DESIGNING FAULT-INJECTION EXPERIMENTS FOR THE RELIABILITY OF EMBEDDED SYSTEMS

Allan L. White, NASA Langley, Hampton, Virginia

Abstract

This paper considers the long-standing problem of conducting fault-injections experiments to establish the ultra-reliability of embedded systems. There have been extensive efforts in fault injection, and this paper offers a partial summary of the efforts, but these previous efforts have focused on realism and efficiency. Fault injections have been used to examine diagnostics and to test algorithms, but the literature does not contain any framework that says how to conduct fault-injection experiments to establish ultra-reliability. A solution to this problem integrates field-data, arguments-from-design, and fault-injection into a seamless whole. The solution in this paper is to derive a model reduction theorem for a class of semi-Markov models suitable for describing ultra-reliable embedded systems. The derivation shows that a tight upper bound on the probability of system failure can be obtained using only the means of system-recovery times, thus reducing the experimental effort to estimating a reasonable number of easily-observed parameters. The paper includes an example of a system subject to both permanent and transient faults. There is a discussion of integrating fault-injection with field-data and arguments-from-design.

Introduction

Background

A long standing problem in the field of ultra-reliable digital control systems is the design of a fault injection experiment for system validation. Such an experiment combines arguments-from-design, field-data-on-fault-occurrence, and results-from-fault-injections. If the system successfully completes the experiment, then the system has a given reliability at a certain confidence level.

Such experiments are often considered impossible. One stated goal is that a flight control system has less than one in a billion chance of failure during a ten hour flight. To establish this at the equivalent confidence level requires 1000 aircraft flying continuously for 21,000 years. Not even a six order magnitude gain in efficiency would make this experiment feasible.

The first response to this problem is to construct a model of the system using our knowledge of system structure and component failure rates, but this approach has the inherent problem of describing system recovery from faults. System recovery is a complex process involving failure modes, application software, diagnostic software, and system architecture. No detail of the model can be arbitrarily omitted since any detail may have a greater effect on the final computation than the small probability to be computed. This has led to more and more complex models that are experimentally intractable: some of the states and transitions in these models are not observable, and the large number of parameters to estimate makes achieving a high confidence level infeasible.

The solution in this paper is to derive a model reduction theorem for a class of semi-Markov models suitable for describing ultra-reliable embedded systems. The theorem shows a tight upper bound on the probability of system failure needs only the means of system recovery times, thus reducing the experimental effort to simple and easily-observed parameters.

The typical reliability model assumes the system works correctly if the components are fault free, and this assumption is often not stated, but we will consider it explicitly because we want to create a seamless integration of arguments-from-design and fault injection. Arguments-from-design proceed by demonstrating that if a system begins in a certain state and receives certain inputs then its outputs will be acceptable. Any such argument requires a straight-forward architecture. If the system is non-reconfigurable, the argument needs to include the presence of faulty components. If the system is reconfigurable, the fault-injection experiment must observe that reconfiguration places the system in an acceptable state. The argument-from-design can

ignore the complex reconfiguration process. The amount of diagnostics established by arguments-from-design can vary with the system. In the example below, arguments from design are expected to cover the first two fault occurrences.

Procedure

The difficult part of a reliability model is a description of system recovery. It is a complex procedure involving propagation of the fault through hardware; interaction of a fault with diagnostic, application, and system programs; comparison voting; and possibly system reconfiguration. In addition, if the required probability of system failure is extremely small, then no detail of system recovery can be arbitrarily ignored since the effect of an omitted element could have more effect on the system than the probability that is to be computed.

These considerations have led to the construction of more and more complicated system recovery models, but there are several problems with this approach. First, there is no guidance for how much detail is needed. Second, detailed models of recovery contain states that are not observable such as the propagation of a hardware fault into an application program. Third, detailed models contain numerous parameters, and obtaining these parameters by experiments would be overwhelming, especially if a high confidence level is sought.

We offer a solution to this problem by deriving a model reduction theorem that yields upper and lower bounds for the probability of system failure in terms of the means and variances of system recoveries where the upper bound uses only the means. These bounds are tight if the system has the desirable properties of low component failure rates and fast system recoveries. Furthermore, the upper bound only uses the means of system recoveries. Since we need only a few, easily observed parameters, experiments are brought within reach.

The procedure in this paper differs from the usual one of (1) constructing a model and (2) computing the probability of failure from the model. This procedure (1) constructs a model, (2) uses the theorem to write a formula for an upper bound for system failure in terms of component failure rates, operating time, and system parameters, (3) conducts experiments to obtain upper confidence bounds for the parameters, and (4) combines the formula and upper-bounds-for-the-parameters to get an upper confidence bound for the probability of system failure.

It is clear from the above that the system must be overbuilt: more reliable than the requirement. If the system has exactly the required probability of failure, then any upper bound will be greater than the required probability.

Faults, Field Data, and Diagnostics

In this paper, a fault is an input-output malfunction of a device. This definition accomplishes two goals. First, it relates fault injection in the lab to the model of fault occurrence in the field since, in the field, a device is declared faulty when it begins producing observable errors. Second, it lends itself to achieving a high diagnostic level which, we will see, is vital for demonstrating ultra-reliability. An approach that has a fault as something that happens inside a device makes a fault both difficult to observe and hard to detect.

The actual fault pattern that appears at the output may be hard to obtain. Hence, part of system design may be some arrangement for the detection of any fault pattern.

It is apparent that a system will be designed and validated for a given class of faults. This class of faults will have to be stated (and agreed upon) in the initial stages of design.

Applicability and Feasibility

From a theoretical point of view, this method of designing experiments is widely applicable. It requires that the system use high quality components with a constant failure rate. System recovery is semi-Markov since the time for system recovery depends on the time since fault occurrence. It also requires a system to have a high diagnostic level which is a characteristic of highly reliable systems.

Feasibility depends on a low overall fault occurrence rate and fast recovery. The experiment gathers data on each fault recovery, and a low fault occurrence rate implies the number of system-recoveries during the operating time will not be too large. Fast system recoveries imply only a small amount of time is spent observing system recoveries in the experiment.

Outline of Paper

The next section contains a literature survey. The results in this paper do not depend on any previous results in the literature, but applying the results in this paper can use the extensive literature on the realistic and efficient simulation of faults. A section on preliminaries covers the miscellaneous topics of confidence level, field data, terminology, and the central limit theorem. The next section presents and derives the model reduction theorem. After the theorem is presented, two sections describe the system and its path space model for the design of the experiment. Since the theorem is based on paths through a model, this last section shows how to handle with a model with loops due to transient faults by unwinding the loops. There are potentially an infinite number of paths, but almost all the paths will have a negligible contribution to system failure because of the small probability of more than a certain number of fault occurrences. (A fault occurrence can include the dependent case of faults in more than one component, and the theorem includes this case, although the hypothetical system only considers single component failures for a simple first example.)

Literature Survey

There is a large body of literature on fault injection, and this section can only offer a cursory description of the efforts. Nevertheless, it is possible to describe what has been done, and then note how this paper differs from previous work. The survey below is by topic. Since most papers discuss several topics, most papers appear more than once.

There are papers that survey the field and/or advocate fault injection as a useful tool [9,14,50,51,53]. One question is authenticity--do laboratory fault injections mimic actual fault occurrences [22, 45, 55]. Some papers use system architecture to design efficient and effective fault injection [1, 4, 6, 8, 20, 21, 24, 25, 26, 27, 29, 30, 31, 39, 40, 42, 44, 46, 47, 48, 49, 55]. Others use the results of fault injections to compare or design systems [4, 8, 11, 12, 23 30, 36, 38, 39, 41, 47, 54]. Fault injections are used to search for design flaws [3, 6, 17, 48]. They are used to test software as well as hardware [8, 13, 28, 50, 51, 52, 53]. There are a number of programs and tools for efficient fault injection [2, 7, 10, 22, 24, 26, 28, 31, 35, 43, 45, 49, 55, 56]. There are efforts to model fault propagation and effect [5, 7, 12, 17, 19, 22, 23, 27, 28, 30, 32, 36, 38, 39, 40, 41, 44, 45, 47, 50, 52]. Coverage is a popular topic, and the meaning of coverage can vary from simple detection to complete system recovery and reconfiguration [1, 2, 3, 4, 5, 10, 11, 12, 15, 16, 17, 18, 20, 21, 27, 29, 31, 33, 36, 37, 38, 39, 40, 41, 42, 43, 44, 54, 55]. There are papers about the efficient estimation of coverage [15, 16, 18, 27, 36, 43], and there are papers about incorporating coverage into a reliability model [5, 13, 47].

This paper is a modest effort, but different from all the above. It considers only hardware. The goal is to derive results in probability that permit establishing ultra reliability (for hardware) with a moderate fault injection effort. This paper is different enough from previous efforts that it is technically self contained.

Preliminaries

Conducting the Experiment

The trials in an experiment consist of injecting certain faults and observing system recovery. Recovery must include detection and identification. If the system is reconfigurable, recovery must include reconfiguration. Any trial can last only a short period of time, and the insistence that the system successfully recovers within this period is a stringent one. This requirement can be relaxed, but at the cost of additional computational effort [57]. As an introductory effort, this paper accepts the simpler, although more demanding, requirement.

From the time of fault injection until recovery, the system is monitored to insure it maintains process control. Acceptable control depends on the environment, but any deviation outside acceptable control is considered system failure.

Overall confidence level

A confidence level is a quantitative measure of the quality of an experiment. If there are random elements present, it is possible for an experiment to mislead us, and the confidence level gives the probability that the experiment has misled us. A 99% confidence level means that there is a 0.99 (or more) chance that the experiment leads us to the correct conclusion. This paper takes the position that the

quality of the experiment should match the importance and quality of what is being established. In the following if the requirement is that the probability of system failure be p (or less), then the experiment is designed to have a confidence level of 100(1-p)%.

Since a reliability model has numerous parameters to be estimated, the final confidence level is a combination of the confidence level for each of the parameters. The result, which does not assume independence, is as follows [58].

Suppose $[\alpha_i , \beta_i]$ is a $100(1-h_i)$% confidence interval for p_i for $1 \le i \le n$, then $([\alpha_1 , \beta_1],..., [\alpha_n , \beta_n])$ is a $100(1- h_1 - ... - h_n)$% confidence interval for $(p_1 ,..., p_n)$.

Hence, the more parameters to be estimated, the higher the confidence level on each must be to maintain an overall high confidence. This is one of the motivators for the model reduction theorem in next section and the reason for introducing integrated-recovery-distributions when designing the experiment.

Confidence Level for Lack of Diagnostics

We wish an upper confidence bound u for the probability that a fault is not detected, which is denoted by (1-D) in the models below. Assuming all faults are detected in n trials, the number of trials needed to establish u as a upper bound at the 100(1 – alpha)% level is

$$(1 – u)^n = \text{alpha} \qquad (1)$$

This can be expanded to solve for n in case some faults are undetected during the experiment by adding more terms in the binomial expansion. For instance, the number of trials n needed to establish that n is an upper bound for (1-D) at the 100(1 – alpha)% level if zero or one faults are undetected is given by

$$(1 – u)^n + n(1 – u)^{n-1} = \text{alpha}. \qquad (2)$$

The upper and lower bounds are derived in section four by considering all the paths from initial states to system-failure states. It is a feature of this path-space approach that for an upper bound on system failure we also need an upper bound for D, the probability that a fault is detected. We will take 1 as a 100% upper confidence bound on D.

Confidence Level for Means of System Recoveries

The upper bound on the probability-of-system-failure uses the averages of the system recoveries, and confidence intervals for averages are derived from the central limit theorem that says a sample average is approximately normally distributed. A problem is that the confidence levels are extremely high, and the normal approximation may or may not be accurate enough even if the sample size is large. This statistical point requires more study, and this is a general problem. There are a few results [58], but the consensus is that "no systematic studies along this direction seem to have been done" [62].

Two Results on Fault Injection

We use two results in probability for fault injection [59]. Suppose components have failure rates $\lambda_1 , \ldots , \lambda_n$. The probability that component j has failed given a component has failed is

$$\lambda_j /(\lambda_1 + \ldots + \lambda_n). \qquad (3)$$

When injecting a fault, the experimenter will allow a time S for system recovery, and if the system does not recovery within that time, it will be declared a system failure. When injecting a double fault, the time of injection is given by the uniform distribution on [0, S].

The Model Reduction Theorem

A reliability model can be regarded as a collection of paths from the initial state (or states) to the failure state (or states). By the semi-Markov property, an arbitrary path can be arranged as in figure 1. At first glance, such an approach might appear to not include transient faults or correlated faults, but transient faults can be handled by unwinding the loops. The unwinding process will end as the probability of more fault occurrences becomes negligible. Correlated faults can be handled by letting a transition represent the failure of more than one component.

In the first line of figure 1, the successful transitions are constant rate processes competing against other constant rate processes. In the second

line the successful transitions are general distributions (system recovery distributions) competing against other general distributions and constant rate processes. In the third line the successful transitions are constant processes competing against general distribution functions and other constant rate processes. For notation

D(T) = Probability of traversing the path in figure 2 by time T

W(T) = Probability of reaching state B_1 by time T

$p(F_i)$ = Probability the transition $dF_{1,i}$ is successful

$\mu(F_i)$ = First conditional moment of $dF_{1,i}$

$\sigma^2(F_i)$ = Conditional variance of $dF_{1,i}$

$\mu(C_j)$ = First moment of the holding time in state C_j

$\sigma^2(C_j)$ = Variance of the holding time in C_j.

The probability W(T) is easy to compute. A convenient approximation which is used in section VI, but whose derivation is left to the reader is

$$\frac{\lambda_1 \cdots \lambda_k \, T^k}{k!} \geq W(T)$$

$$\geq \frac{\lambda_1 \cdots \lambda_k \, T^k}{k!} \left[1 - \frac{T \sum_{i=1}^{k} (\lambda_i + \gamma_i)}{k+1} \right] \quad (4)$$

Let

$$\Delta = \left[\mu(F_1)T\right]^{1/2} + \ldots + \left[\mu(F_m)T\right]^{1/2} + \left[\mu(C_1)T\right]^{1/2} + \ldots + \left[\mu(C_n)T\right]^{1/2} \quad (5)$$

Assume

$$\mu(F_1), \ldots, \mu(F_m), \ldots, \mu(C_1), \ldots, \mu(C_n) < 1 \quad (6)$$

and $\Delta < 1$.

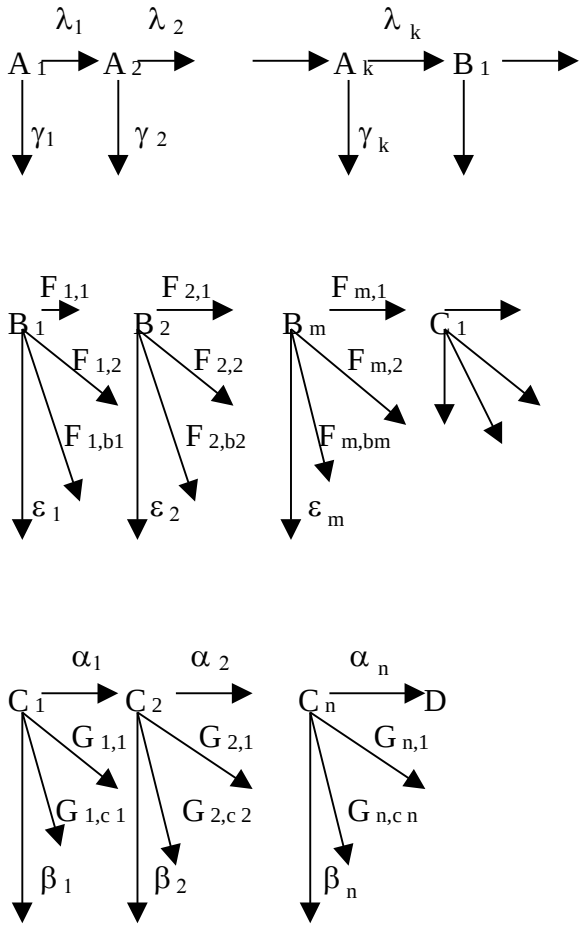

Figure 1. Arbitrary path in a semi-Markov model

The proof uses two standard results in probability. If H is a distribution such that H(0-) = 0, then

$$\int_0^\infty [1 - H(t)]dt = \int_0^\infty t\, dH(t) = \mu(H) \qquad (7)$$

$$\int_c^\infty dH(t) \le \frac{\mu^2(H) + \sigma^2(H)}{c^2} \quad \text{for } c > 0 \qquad (8)$$

Theorem: With the assumptions and notation as above, upper and lower bounds for reaching state D in figure 1 by time T are

$$\begin{aligned} & W(T) \prod_{i=1}^{m} p(F_i) \prod_{j=1}^{n} \alpha_j\, \mu(C_j) \\ & \ge D(T) \\ & \ge W(T - \Delta) \qquad (9) \\ & \prod_{i=1}^{m} p(F_i) \left[1 - \frac{\mu^2(F_i) + \sigma^2(F_i)}{\mu(F_i)\, T} \right] \\ & \prod_{j=1}^{n} \alpha_j \left[\mu(C_j) - \frac{\mu^2(C_j) + \sigma^2(C_j)}{\mu(C_j)\, T} \right] \end{aligned}$$

Proof of the theorem

Let q(t) be the density function for the probability of reaching state B_1 of figure 1. The derivation is easier to follow if we assume the recovery distributions have density functions, and we will use f(x) dx instead of dF(x) although the theorem holds in the general case.

$$\begin{aligned} D(T) = \int_0^T q(t) \\ & \int_0^{T-t} e^{-\varepsilon_1 x_1} \left[1 - F_{1,2}(x_1)\right] \cdots \left[1 - F_{1,b_1}(x_1)\right] f_{1,1}(x_1) \\ & \qquad \vdots \\ & \int_0^{T-t-x_1-\cdots x_{m-1}} e^{-\varepsilon_m x_m} \\ & \qquad \left[1 - F_{m,2}(x_m)\right] \cdots \left[1 - F_{m,b_m}(x_m)\right] f_{m,1}(x_m) \end{aligned}$$

$$\begin{aligned} & \int_0^{T-t-x_1-\cdots x_m} \alpha_1 e^{-\alpha_1 y_1} e^{-\beta_1 y_1} \\ & \qquad \left[1 - G_{1,1}(y_1)\right] \cdots \left[1 - G_{1,c_1}(y_1)\right] \\ & \qquad \vdots \\ & \int_0^{T-t-x_1-\cdots x_m-y_1-\cdots y_{n-1}} \alpha_n e^{-\alpha_n y_n} e^{-\beta_n y_n} \\ & \qquad \left[1 - G_{n,1}(y_n)\right] \cdots \left[1 - G_{n,c_n}(y_n)\right] \\ & \qquad dy_n \cdots dy_1\, dx_m \cdots dx_1\, dt \qquad (10) \end{aligned}$$

Working with just the limits of integration

$$\begin{aligned} & \int_0^{T-\Delta} \int_0^{r_1} \cdots \int_0^{r_m} \int_0^{s_1} \cdots \int_0^{s_n} \le D(T) \\ & \le \int_0^{T} \int_0^{\infty} \cdots \int_0^{\infty} \int_0^{\infty} \cdots \int_0^{\infty} \qquad (11) \end{aligned}$$

where

$$r_i = \left[\mu(F_i)\, T\right]^{1/2}; \quad s_i = \left[\mu(C_j)\, T\right]^{1/2} \qquad (12)$$

$$\Delta = r_1 + \cdots + r_m + s_1 + \cdots + s_n \qquad (13)$$

It remains to establish four inequalities ((14) through (17) below).

$$\int_0^{\infty} e^{-\varepsilon_i x_i} \left[1 - F_{i,2}(x_i)\right] \cdots \left[1 - F_{i,b_i}(x_i)\right] dF_{i,1}(x_i)$$
$$= p(F_i) \qquad (14)$$

$$\int_0^{r_i} e^{-\varepsilon_i x_i} \left[1 - F_{i,2}(x_i)\right] \cdots \left[1 - F_{i,b_i}(x_i)\right] dF_{i,1}(x_i)$$
$$= \int_0^{\infty} e^{-\varepsilon_i x_i} \left[1 - F_{i,2}(x_i)\right] \cdots \left[1 - F_{i,b_i}(x_i)\right] dF_{i,1}(x_i) - \frac{p(F_i)}{p(F_i)}$$
$$\int_{r_i}^{\infty} e^{-\varepsilon_i x_i} \left[1 - F_{i,2}(x_i)\right] \cdots \left[1 - F_{i,b_i}(x_i)\right] dF_{i,1}(x_i)$$
$$\geq p(F_i)\left[1 - \frac{\mu^2(F_i) + \sigma^2(F_i)}{r_i^2}\right] \qquad (15)$$

$$\int_0^{\infty} \alpha_j e^{-\alpha_j y_j} e^{-\beta_j y_j} \left[1 - G_{j,1}(y_j)\right] \cdots \left[1 - G_{j,c_j}(y_j)\right] dy_j$$
$$= \alpha_j \mu(C_j) \qquad (16)$$

$$\int_0^{s_j} \alpha_j e^{-\alpha_j y_j} e^{-\beta_j y_j} \left[1 - G_{j,1}(y_j)\right] \cdots \left[1 - G_{j,c_j}(y_j)\right] dy_j$$
$$= \alpha_j \int_0^{\infty} e^{-\alpha_j y_j} e^{-\beta_j y_j} \left[1 - G_{j,1}(y_j)\right] \cdots \left[1 - G_{j,c_j}(y_j)\right] dy_j$$
$$- \alpha_j \int_{s_j}^{\infty} \left\{ e^{-\alpha_j y_j} e^{-\beta_j y_j} \left[1 - G_{j,1}(y_j)\right] \cdots \left[1 - G_{j,c_j}(y_j)\right] \right\} dy_j$$
$$\geq \alpha_j \left[\mu(C_j) - \frac{\mu^2(C_j) + \sigma^2(C_j)}{s_j^2} \right] \qquad (17)$$

The theorem is proved by substituting these inequalities into the previous inequalities for D(T).

Description of the Nonreconfigurable Sevenplex Example

General Description

The architecture is a nonreconfigurable sevenplex where each module consists of a computer-on-a-chip plus six transmission lines to the other modules. There are seven computers and forty two links. The requirement is to establish that there is less than one chance in a billion of failure during a ten hour flight, and to establish this at the 100(1-1e-9)% confidence level. The permanent and transient failure rates for the computers are 1e-6/hour and 1e-5/hour respectively. The rates for the links are 1e-5/hour and 1e-4/hour.

Masking, Detection, and Identification

The general principles for masking arbitrary faults are: (1) 3k+1 components are needed to reach consensus in the presence of k (arbitrarily malicious) faults and (2) any message from a good component can be identified as being from that component [63, 64].

For both diagnostics and Byzantine resilience, this sevenplex has the following features.

(i) The computational and decision making components are computers-on-a-chip. The faults (input-output malfunctions) appear on the output registers, and any incorrect output will be transmitted to another computer.

(ii) Communication is point-to-point for all the computers. Hence, any good computer knows from which computer it received a correct message. It can, however, require some time to determine if the incorrect message arises from a faulty computer or a faulty link.

Hence, we assume that arguments-from-design have established the following.

The system can tolerate faulty components as long as there are no more than two faulty components currently in the system.

In addition, faults are characterized as input-output malfunctions in order that fault injection in the lab corresponds to observed fault occurrence in the field. Since the total connectivity of the system conveys any malfunction to the other processors, the good processors can detect any fault occurrence. As long as there are five good processors, the good processors can identify the source of the fault with one exception. If processor A tells the other processors it has received a faulty message from processor B and the other processors have received a correct message from processor B, then the likely culprit is the link between A and B. It is possible, however, that processor A has sent a malicious message. For this reason, processor A is also suspect, and both processor A and the link will have to be examined during the maintenance check. During the experiment, declaring the link as faulty will be considered a successful recovery. During the experiment and run-time, the most damage such a malicious processor can do is shut down all the links to it.

The arguments have not established the time it takes to detect and identify the faulty units, nor do they make any assertions about system behavior if three or more faults are present. This part belongs to the experimental effort.

The Path Space Model and Design of the Experiment

Outline

The construction of the model and the design of the experiment are intertwined. There are three steps.

1. Preliminary experiments are conducted to get initial estimates of the model parameters.

2. These initial estimates are used to construct a model.

3. The model is used to determine the number and types of fault injections required.

Preliminary Experiments

Assume the preliminary values for the mean and variance of system recovery are given in the first three columns of Table 2. Recovery from a computer transient takes longer because of the need to rewrite the internal stored values. The model will be constructed and the experiment designed on the assumption that the initial estimates are reasonably accurate. If these estimates are too large, then the model and experiment will be inefficient. If these parameters are too small, then the model and experiment will not be adequate.

This discussion assumes, for convenience, that the final estimates match the initial estimates.

Integrated Recovery Functions

The typical model has a different recovery function for each type of fault, but this creates a proliferation of parameters, and we have seen in the subsection on combining confidence levels that this increases the number of trials needed to maintain a high overall confidence level. Hence, each recovery function in the model below integrates recovery from several types of faults although a distinction is made between permanent and transient faults since permanent faults remain in the system and additional data must be collected when there are three or more faults in the system.

Recovery integration is handled by proportional sampling. For instance, recovery H_1 in figure 2 handles both processor and link permanent faults. Suppose λ is the failure rate for processors, φ is the failure rate for links, and N is the number of faults to be injected to obtain the parameters for H_1.

Then the experiment will inject $7\gamma/(7\gamma + 42\varphi)$ N processor faults and $42\varphi/(7\gamma + 42\varphi)$ link faults.

Recovery H_2 handles a permanent fault followed by a transient fault. With the notation of γ for processor-rate, φ for link-rate, p for permanent, and t for transient, the proportions are given in table 1.

Table 1. Proportion of Injected Faults for Recovery Distribution H_2

Type	Proportion	
Processor-Processor	$\frac{7\gamma_p}{7\gamma_p + 42\varphi_p}$	$\frac{6\gamma_t}{6\gamma_t + 42\varphi_t}$
Processor-Link	$\frac{7\gamma_p}{7\gamma_p + 42\varphi_p}$	$\frac{42\varphi_t}{6\gamma_t + 42\varphi_t}$
Link-Processor	$\frac{42\varphi_p}{7\gamma_p + 42\varphi_p}$	$\frac{7\gamma_t}{7\gamma_t + 41\varphi_t}$
Link-Link	$\frac{42\varphi_p}{7\gamma_p + 42\varphi_p}$	$\frac{41\varphi_t}{7\gamma_t + 41\varphi_t}$

Description and Figures for the Model

Based on the preliminary results from exploratory fault injections and the accompanying calculations, the system model was constructed, and the first part of this model is shown in figure 2 The construction of this model illustrates the difference between constructing a conservative model for ease of experimentation and constructing a more accurate model for a more precise calculation of system failure.

The model begins in state 1 and transitions to state 2 with a permanent fault and to state 17 with a transient fault which is indicated by a dashed arrow. The type of permanent fault injected is chosen randomly from a multinomial distribution according to the ratio as described in the previous section. These fault injections yield a mean and variance for H_1, the recovery distribution for the first permanent fault that occurs.

An occurrence of a permanent fault while recovering takes the system to state 3. Since this is expected to be a rare occurrence, the model ignores system recovery although the system can tolerate two faults. The occurrence of a third fault in state 3 is considered a system failure F_1. The occurrence of a transient fault in stated 2, however, has a probability large enough that the model must track it – to state 4 where H_2 is the recovery model for both faults. A third fault occurrence in system 4 is unlikely enough that it will be considered system failure. A successful recovery from state 4 takes the system to state 5 where a permanent is treated similarly to a permanent failure in state 2 and a transient failure takes the system to the recovery state 7. In state 7, any fault occurrence is taken as a system failure. Recovery goes to state 8 where there is one permanent fault in the system. In state 8, any additional faults are unlikely enough that system recovery is not tracked and two additional faults place the system in failure state F_5.

Recovery from the first permanent fault takes the system to state 10 where a permanent fault leads to state 11 (and a transient fault to state 58 although this part of the model is not shown). In state 11, any failure that occurs during system recovery is considered a system failure. System recovery goes to state 12. Since there are now two faults in the system, we are no longer guaranteed the system will correctly handle a fault. A correctly-handled permanent fault takes the system to state 13 where the recovery process is ignored and an additional fault is declared system failure. A correctly-handled transient fault takes the system to state 14. In state 12, an incorrectly-handled fault takes the system to failure state F_7. The model is simplified and the probability of failure bounded above by having the transition to F_7 use ω, the sum of all failure rates.

In state 14, the occurrence of any fault during system recovery is declared a system failure. Recovery goes to state 15 where there are two faults in the system. The occurrence of a fault that is not correctly handled (not detected) takes the system to a failure state. The occurrence of a fault that is correctly handled takes the system to state 16 where system recovery is ignored and another fault occurrence is considered system failure. The rest of the model is similar. The complete model consists of 69 operational states plus 42 failure states. Because of a lack of space only the first 16 operational states and 11 failure states are displayed in figure 2. The

dotted lines indicate transitions to states not included in this first part.

Fault Injection for Non-recovery

Non-recovery refers to the improper handling of a fault which could range from non-detection to removing a good component to system crash. In the complete model, there are six failure states for improper handling of a fault with two of them present in the model in figure 1: F_7 and F_{10}. In the previous states (12 and 15), the system has two faults present and cannot be guaranteed to handle the third fault correctly. The algebraic upper bounds for these two states are

$$F_7 < \frac{T^3}{6} \propto^2 \omega\,(1-\pi_7) \qquad (18)$$

$$F_{10} < \frac{T^4}{24} \propto^2 \beta\,\omega\,(1-\pi_{10}) \qquad (19)$$

An upper bound of 1e-3 for $(1-\pi_7)$ gives an upper bound for F_7 of 1.43e-10, and the upper of 1e-3 can be established at the 100(1 – 2e-10)% confidence level with 22,322 successful trials. An upper bound of 1e-2 for $(1-\pi_{10})$ gives an upper bound of 1e-11 for F_{10}, and the upper bound of 1e-2 can be established at the 100(1 – 1e-11)% confidence level with 2,520 successful trials.

It's not shown because of a lack of space, but a conditional probability of non-recovery of 1e-2 is also sufficient for the other four non-recovery failure transitions, and as before this requires 2,520 successful trials for each of them to give each a confidence level of 100(1 – 1e-11)%. The total contribution of the non-recovery states to system failure is 1.89e-10, and the stated trials establish this at the 100(1 – 2.5e-10)% confidence level.

Fault Injection for Recovery Distributions

In the complete model, there are 22 recovery functions whose means must be estimated. The first five recovery distributions are displayed in figure 2. The computations actually use the upper bounds for the means which need to be estimated at some confidence level. If each of the 22 upper bounds are estimated at the 100(1 – 3e-11)% level, then the total contribution to the lack of confidence for the distribution functions is 6.6e-10. Combined with the lack of confidence for non-recovery of 1.89e-10, this gives an overall confidence level for the experiment of 100(1 – 8.49e-10)% which satisfies the desired goal of a 100(1 – 1e-9)% level.

As mentioned before, there is a paucity of results about the tails of the normal approximation. Hence, we will derive a conservative upper-bound for the tails of the normal. We begin by bounding the normal density function above with

$$f(x) = \frac{1}{\sqrt{2\,\pi}}\; x \exp(-\frac{x^2}{2}) \qquad (20)$$

when x > 1. We have

$$\int_{6.83}^{\infty} f(x) \approx 3e-11 \qquad (21)$$

which says that a sample-mean plus 6.83 sample-standard-deviations gives an upper confidence bound of 100(1 – 3e-11) % .

Continuing to be conservative, we will use the estimated-mean plus the estimated-standard-deviation as an upper bound for the mean. If there are 10,000 trials, the standard deviation of the estimator is one-hundredth of the population-standard-deviation. Hence, we are using the sample-average plus 100 sample-standard-deviations (instead of 6.83 of them).

Table 2 displays the estimated means and standard deviations in seconds for the five system recoveries depicted in figure 2. Recoveries that handle transients take longer because of system restoration. Table 3 displays the eleven failure states in figure 2. The upper bound computations use the upper confidence bounds for the means which is the estimated means plus the estimated standard deviations.

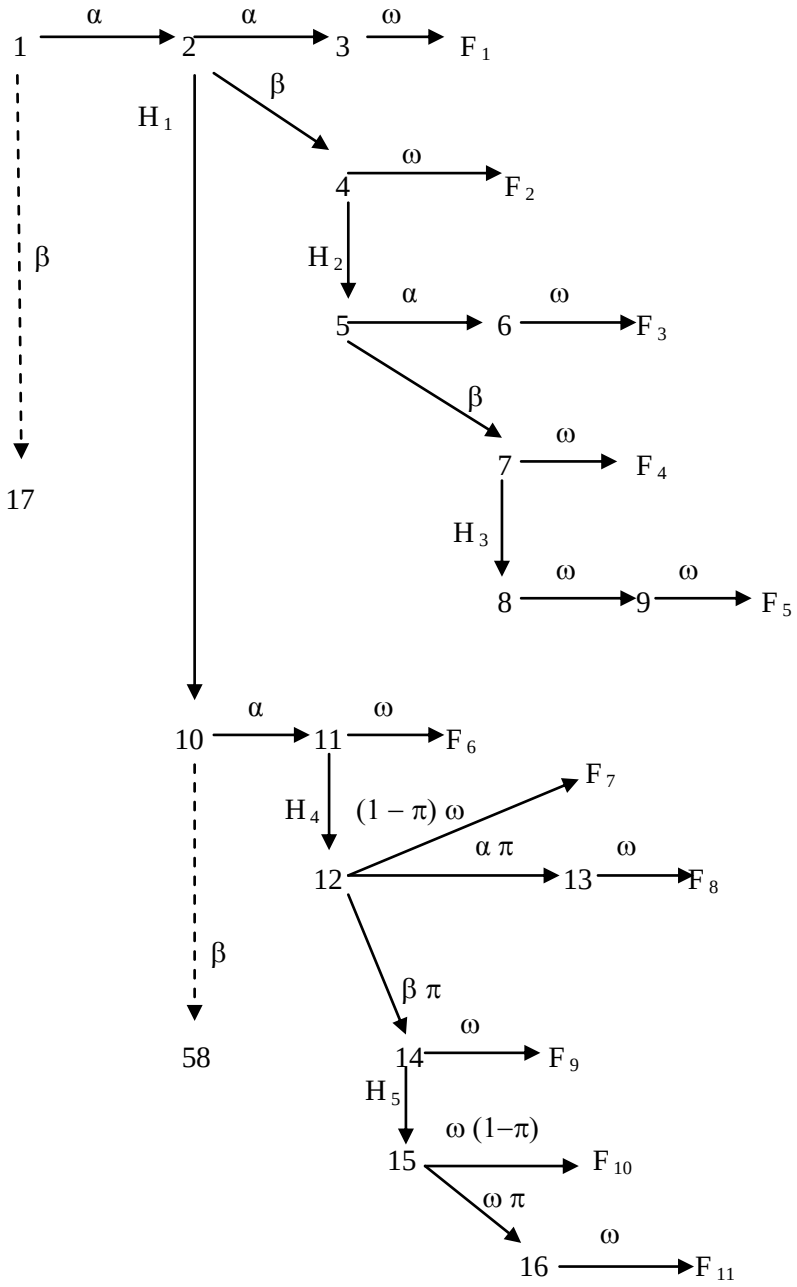

Figure 2. First Part of the path Space Model for the Sevenplex

Table 2. Estimates for Recovery Distributions

System Recovery	Estimated Population Mean	Estimated Population Standard Deviation
H_1	2	2
H_2	6	4
H_3	4	2
H_4	2	2
H_5	4	2

Table 3. First Eleven Failure States

State	Algebraic Upper Bound	Value
F_1	$\frac{T^2}{2} \propto^2 \omega\, \mu(H_1)$	4.76e-11
F_2	$T \propto \beta\, \omega\, \mu(H_1)\, \mu(H_2)$	2.64e-13
F_3	$\frac{T^3}{6} \propto^2 \beta\, \omega\, \mu(H_1)$	6.77e-13
F_4	$\frac{T^2}{2} \propto \beta^2\, \omega\, \mu(H_1)\, \mu(H_3)$	3.39e-15
F_5	$\frac{T^4}{24} \propto \beta^2\, \omega^2\, \mu(H_1)$	7.95e-14
F_6	$\frac{T^2}{2} \propto^2 \omega\, \mu(H_4)$	4.76e-11
F_7	$\frac{T^3}{6} \propto^2 \omega\, (1-\pi_7)$	1.43e-10
F_8	$\frac{T^4}{24} \propto^3 \omega\, \mu$	1.52e-10
F_9	$\frac{T^3}{6} \propto^2 \beta\, \omega\, \mu(H_5)$	1.02e-12
F_{10}	$\frac{T^4}{24} \propto^2 \beta\, \omega\, (1-\pi_{10})$	1.52e-11
F_{11}	$\frac{T^5}{120} \propto^2 \beta\, \omega^2$	1.43e-11

Summary for Fault Injections

Summing all the upper bounds for all the failure states gives 5.99e-10. The total number of fault injections required is 257,442. If successful, the experiment has established the probability of failure is less than 1e-9 at an equivalent confidence level.

Summary

We approach the problem of designing fault-injection experiments by reducing the number of parameters to be estimated. The major result is a bound on the probability of system failure in terms of the means and variances of the recovery distributions. The upper bound uses only the means, and this bound is tight if the component failure rates are low and system recovery is fast. Another technique is integrated system recoveries where a single recovery distribution describes the system's reaction to several types of faults. We derive the bounds and apply the techniques to the design of an experiment for a redundant system. A number of problems remain: collection of field data, the accuracy of the normal approximation for extremely high confidence levels, and fault-identification in a Byzantine scenario.

References

[1] Z. Alkhalifa, V. Nair, N. Krishnamurthy, J. Abraham, "Design and evaluation of system-level checks for on-line control flow error detection," IEEE Transactions on Parallel and Distributed Systems, volume 10 Issue 6 (1999), pp. 627-641.

[2] J. Ariat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins, D. Powell, "Fault injection for dependability validation: a methodology and some applications," IEEE transactions on Software Engineering, Volume 16 Issue 2 (1990), pp. 166-182.

[3] J. Ariat, M. Aguera, Y. Crouzet, J.-C. Fabre, E. Martins, D. Powell, "Experimental evaluation of the fault tolerance of an atomic multicast system," IEEE Transactions on Reliability, Volume 39 Issue 4 (1990), pp. 455-467.

[4] J. Ariat, J. Boue, Y. Crouzet, "Validation-based development of dependable systems," IEEE Micro, Volume 19 Issue 4 (1999), pp. 66-79.

[5] J. Ariat, A. Costes, Y. Crouzet, J.-C. Laprie, D. Powell, "Fault injection and dependability evaluation of fault-tolerant systems," IEEE Transactions on Computers, Volume 42 Issue 8 (1993), pp. 913-923.

[6] D. Avreesky, J. Arlat, J.-C. Laprie, Y. Crouzeet, "Fault injection for formal testing of fault tolerance," IEEE Transactions on Reliability, Volume 45 Issue 3 (1996), pp. 443-455.

[7] J. Barton, E. Czek, Z. Segall, D. Siewiorek, "Fault injection experiments using FIAT," IEEE Transactions on Computers, volume 39 Issue 4 (1990), pp. 575-582.

[8] A. Brombacher, I. van Beurdeu, "RIFIT: analyzing hardware and software in safeguarding systems," Reliability Engineering & System Safety, Volume 66 Issue 2 (1999), pp. 149-156.

[9] J. Carreira, D. Costa, J. Silva, "Fault injection spot-checks computer system dependability," IEEE Spectrum, Volume 36 Issue 8 (1999), pp 50-55.

[10] P. Cascaval, S. Bennett, "Efficient march test for 3-coupling faults in random access memories," Microprocessors and Microsystems, Volume 24 Issue 10 (2001), pp. 501-509.

[11] P. Cheynet, R. Velazco, S. Rezgui, L. Peters, K. Beck, R, Ecoffet, "Digital fuzzy control: a robust alternative suitable for space application," IEEE Transactions on Nuclear Science, Volume 45 Issue 6 (1998), pp. 2941-2947.

[12] G. Choi, R. Iyer, V. Carreno, "Simulated fault injection: a methodology to evaluate fault tolerant microprocessor architectures," IEEE Transactions on Reliability, Volume 39 Issue 4 (1990), pp. 486-491.

[13] J. Choi, P. Seong, "Dependability estimation of a digital system with consideration of software masking effects on hardware faults," Reliabiity Engineering & System Safety, volume 71 Issue 1 (2001), pp. 45-55.

[14] J. Clark, D. Pradhan, "Fault injection: a method for validating computer-system dependability," Computer, Volume 28 Issue 6 (1995), pp. 47-56

[15] C. Constantinescu, "Using multi-stage and stratified sampling for inferring fault-coverage probabilities," IEEE transactions on Reliability, Volume 44 Issue4 (1995), pp. 632-639.

[16] C. Constantinescu, "Inferring coverage probabilities by optimum 3-stage sampling," Microelectronics and Reliability, Volume 37 Issue 8 (1998), page 1280.

1[7] C. Constantinescu, "Teraflops supercomputer: architecture and validation of the fault tolerant mechanisms," IEEE Transactions on Computers, Volume 49 Issue 9 (2000), pp. 886-894.

[18] M. Cukier, D. Powell, J. Ariat, "Coverage estimation methods for stratified fault-injection," IEEE Transactions on Computers, Volume 48 Issue 7 (1999), pp. 707-723.

[19] E. Czeck, D. Siewiorek "Observations on the effects of fault manifestation as a function of workload," IEEE transactions on Computers, Volume 41 Issue 5 , (1992), pp. 559-566.

[20] M. Dalpasso, M. Favalli, P. Olivo, B. Ricco, "Fault simulation of parametric bridging faults in CMOS IC's," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 12 Issue 9 (1993). pp. 1403-1410.

[21] T. Delong, B. Johnson, J. Profeta, "A fault injection technique for VDL behavioral-level models," IEEE Design & Test of Computers, Volume 13 Issue 4 (1996), pp. 24-33.

[22] K. Goswami, "DEPEND: a simulation-based environment for system level dependability analysis," IEEE Transactions on Computers, Volume 46 Issue 1 (1997), pp. 60-74.

[23] J. Hlavicka, S. Racek, P. Herout, "Evaluation of process controller fault tolerance using simulation," Simulation Practice and Theory, volume 7 Issue 8 (2000), pp. 769-790.

[24] G.-H. Hwang, W.-Z. Shen, "Fault analysis and automatic test patttern for break faults in programmable logic arrays," IEE Proceeding-Circuits Devices and Systems, Volume 143 Issue 3 (1996), pp. 157-166.

[25] S.-A. Hwang, J.-H. Hong, C.-W Wu, "Sequential circuit fault simulation using logic emulation," IEEE Transactions on Computer-

Aided Design of Integrated Circuits and Systems, Volume 17 Issue 8 (1998). Pp. 724-736.

[26] Hyung Ki Lee, Dong Sam Ha, "HOPE: an efficient parallel fault simulator for synchronous sequential circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 15 Issue 9 (1996), pp. 1048-1058.

[27] Z. Kaibarczyk, R. Iyer, G. Ries, J. Patel, M. Lee, Y. Xiao. "Hierarchical simulation approach to accurate fault modeling for system dependability," IEEE Transactions on Software Engineering, Volume 25 Issue 5 (1999), pp. 619-632.

[28] W. Kao, r. Iyer, D. Tang, "FINE: A fault injection and monitoring environment for tracing the UNIX system behavior under faults," IEEE Transactions on Software Engineering, Volume 19 Issue 11 (1993), pp. 1105-1118.

[29] J. Karlsson, P. Liden, P. Dahlgren, R. Johanson, U. Gunneflo, "Using heavy ion radiation to validate fault-handling mechanisms," IEEE Micro, Volume 14 Issue 1 (1994), pp. 8-23.

[30] H. Kerkhoff, H. Speck, "Defect-oriented testing of Josephson logic circuits and systems," Physica C: Superconductivity, Volume 350 Issues 3-4 (2001), pp. 261-268.

[31] Kwang-Ting Cheng "Transition fault testing for sequential circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 12 Issue 12 (1993), pp. 1971-1983.

[32] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, "Delayed Intrnet routing convergence," IEEE/ACM Transactions on Networking, Volume 9 Issue 3 (2001), pp. 293-306.

[33] R. Lettner, M. Prammer, C. Scherrer, A. Steininger, "Assessment of computer fault toleranceóa fault injection toolset and the rationale behind it," Computer Standards & Interfaces, Volume 21 Issue 4 (1999), pp. 357-369.

[34] M. Hsueh, T. Tsai, R. Iyer, "Fault injection techniques and tools," Computer, Volume 30 Issue 4 (1997), pp. 75-82.

[35] M. Meyer, R. Camposano, "Active timing multilevel fault simulator with switch-level accuracy," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, volume 14 Issue 10 (1995), pp. 1241-1256.

[36] G. Miremadi, J. Torin, "Evaluating processor-behavior and three error-detection mechanisms using physical fault-injection," IEEE Transactions on Reliability, Volume 44 Issue 3 (1995), pp. 441-454.

[37] D. Powell, E. Martins, J. Ariat, Y. Crouzet, "Estimators for fault tolerance coverage evaluation," IEEE Transactions on Computers, Volume 44 Issue 2 (1995), pp. 261-274.

[38] J. Rajski, J. Tyzer, "The analysis of digital integrators," IEEE Transactions on Computers, Volume 42 Issue 6 (1992), pp. 643-650.

[39] J. Rajski, J. Tyzer, "Accumulator-based compaction of test responses," IEEE Transactions on Circuits and Systems II, Volume 39 Issue 5 (1992), pp. 293-301.

[40] J. Rajski, J. Tyzer, "Test responses compaction in accumulators with rotate carry adders," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 12 Issue 4 (1993), pp. 531-539.

[41] H. Seungjae, K. Shin, "Experimental evaluation of behavior-based failure-detection schemes in real-time communication networks," IEEE Transactions on Parallel and Distributed Systems, Volume 10 Issue 8 (1999), pp. 613-625.

[42] G. Silberman, I. Spillinger, "Using functional fault simulation and the difference fault model to estimate implementation fault coverage," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 9 Issue 12 , (1990). Pp. 1335-1343.

[43] D. Smith, B. Johnson, N. Andianos, J. Profeta, "A variance reduction technique via fault-expansion for fault-coverage estimation," IEEE Transactions On Reliability, Volume 46 Issue 3 (1997), pp. 366-374.

[44] D. Smith, B. Johnson, J. Profeta, ìSystem dependability valuation via a fault list generation algorithm," IEEE Transactions on Computers, Volume 45 Issue 8 (1996), pp. 974-979.

[45] D. Stort, G. Ries, M. Hsueh, R. Iyer, "Dependability analysis of a high-speed network using software-implemented fault injection and simulated fault injection," IEEE Transactions on Computers, Volume 47 Issue 1 (1998), pp. 108-119.

[46] N. Stressman, B. Vinnakota, R. Harjani, "System-level design for test of fully differential analog circuits," IEEE Journal of Solid-State Circuits, Volume 31 Issue 10 (1996), pp. 1526-1534.

[47] C. Stroud, "Reliability of majority voting based VLSI fault-tolerant circuits," IEEE Transactions on Very Large Scale Integration, Volume 2 Issue 4 (1994), pp. 516-521.

[48] A. Tomita, K. Sakamura, "Improving design dependability by exploiting an open model based specification," IEEE Transactions on Computers, volume 48 Issue 1 (1999), pp.24-37.

[49] T. Tsai, M. Hsueh, H. Zhao, Z. Kaibarczyk, R. Iyer, "Stress-based and path-based fault injection," IEEE Transactions on Computers, Volume 48 Issue 11 (1999), pp. 1183-1201.

[50] J. Voas, "Fault injection for the masses," Computer, Volume 30 Issue 12 (1997), pp. 129-130

[51] J. Voas, "Certifying software for high-assurance environments," IEEE Software, Volume 16 Issue 4 (1999), pp. 48-54.

[52] J. Voas, F. Charron, G. McGraw, K. Miller, M. Friedman, "Predicting how badly 'good' software can behave," IEEE Software, Volume 14 Issue 4 (1997), pp. 73-83.

[53] J. Voas, G. McGraw, L. Kassab, L. Voas, "A 'crystal ball' for software liability," Computer, Volume 30 Issue 6 (1997), pp.29-36

[54] C. Walter, "Evaluation and design of an ultra-reliable distributed architecture for fault tolerance," IEEE transactions on Reliability, Volume 39 Issue 4 (1990), pp. 492-499.

[55] C. Yount,, D. Siewiorek, "A methodology for the rapid injection of transient hardware errors," IEEE Transactions on Computers, Volume 45 Issue 8 (1996), pp. 881-891.

[56] M. Zwolinski, "A technique for transparent fault injection and simulation in VHDL," Microelectronics Reliability, Volume 41 Issue 6 (2001), pp. 797-804.

[57] A. White, "Reliability with imperfect diagnostics," Journal Microelectronics and Reliability, Volume 24 Issue 6 (1984), pp. 1069-1076.

[58] S. Wilks, Mathematical Statistics, Wiley, New York, 1963.

[59] S. Karlin and H. Taylor, A First Course in Stochastic Processes, Academic Press, New York, 1975.

[60] Jean-Claude Laprie, "Dependable computing and fault tolerance: concepts and terminology," Proceedings of FTCS-15, 1985, pp.2-11.

[61] A. Friedman, Foundations of Modern Analysis, Dover, New York, 1982.

[62] Anirbon DasGupta, Asymptotic Theory of Statistics and Probability, Springer, New York, 2008.

[63] L. Lamport, R. Shostak, M.Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems 4 (1982), pp. 382–401.

[64] M.Pease, R. Shostak, L. Lamport, "Reaching Agreement in the Presence of Faults," *Journal of the ACM 27* (1980), pp. 228-234.