NASA Technical Memorandum 89058

A Preliminary Transient-Fault Experiment on the SIFT Computer System

Ricky W. Butler and Carl R. Elks

February 1987

(NASA-TM-89058) A PRELIMINARY TRANSIENT-FAULT EXPERIMENT ON THE SIFT COMPUTER SYSTEM (NASA) 43 p CSCL 09B

N87-20770

Unclas
G3/62 45359

NASA
National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

INTRODUCTION

A reconfigurable computer system must distinguish between transient faults and permanent faults. A transient fault usually only causes incorrect behavior temporarily, and consequently the operating system should not permanently remove the affected component via reconfiguration. Unfortunately, transient faults appear to occur more frequently than permanent faults. The available empirical data show that transients occur about 10 times more frequently than permanent faults. (See ref. 1.) Thus, if an operating system removes too many processors affected by transient faults, then the reliability will be seriously compromised. The development of an effective transient/permanent fault discrimination algorithm is a critical problem for fault-tolerant computer system designers. The objective of this experiment is threefold:

1. To gain some fundamental information concerning error latency and the error propagation process in the presence of injected transient faults

2. To obtain the necessary data to perform a reliability analysis of the SIFT computer system (ref. 2) including the effects of permanent and transient faults

3. To determine the effectiveness of the operating system's ability to discriminate between transient and permanent faults

Only a small number of injections have been performed, therefore, statistically significant conclusions cannot yet be drawn. The purpose of this paper is to present the experimental approach and data analysis techniques in detail.

SYMBOLS

W	random variable representing the duration of transient faults
Z^*	random variable representing the elapsed time from fault injection until last error appears.
R^*	random variable representing the elapsed time from fault injection until the system reconfigures
Z	random variable representing the elapsed time from fault injection until last error appears given that *reconfiguration* does not occur
R	random variable representing the elapsed time from fault injection until the system reconfigures given that reconfiguration occurs
λ_T	arrival rate of transient faults
λ_P	arrival rate of permanent faults
$F_W(w)$	distribution of W
$F_{Z^*}(z)$	distribution of Z^*
$F_Z(z)$	distribution of Z
$F_{R^*}(r)$	distribution of R^*
$F_R(r)$	distribution of R
$F_L(t)$	distribution of fault latency
$F_P(t)$	distribution of permanent fault reconfiguration time
$F_{R\|W}(r,w)$	conditional distribution of R given W

$F_{Z|W}(r,w)$ conditional distribution of Z given W

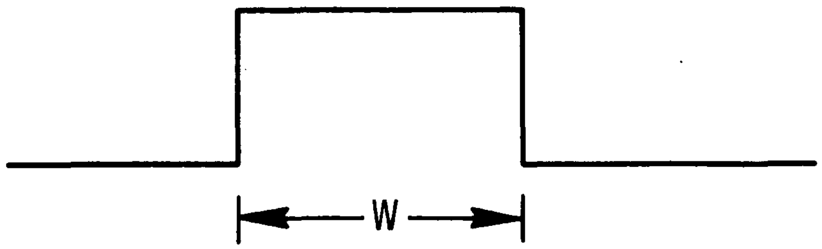
E[.] expected value operator

μ(.) mean of a distribution

σ^2(.) variance of a distribution

EXPERIMENTAL APPROACH

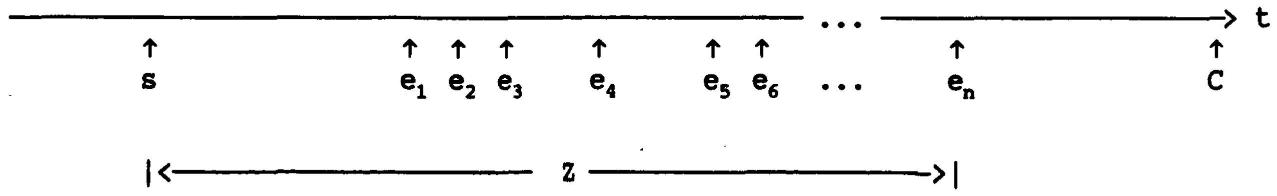
In this experiment, transient faults with a particularly simple waveform are injected:



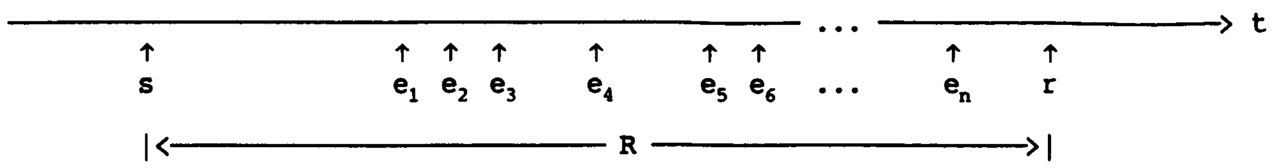
Clearly, there are an infinite number of possible transient waveforms. Since nobody knows what the characteristics of transient waveforms are in nature, we are beginning with this simple waveform. The fault is held active (either stuck-at-1 or stuck-at-0) for W microseconds.

A transient fault may or may not generate errors which are detectable by the operating system's voters. The following two time-graphs illustrate the two possible effects of a transient fault:

Case 1: Reconfiguration does not occur



Case 2: Reconfiguration occurs



where

s = time fault injection initiated

e_i = the time of detection of the ith error $(1 \leq i \leq n)$

r = time operating system reconfigures

C = censoring point (i.e. point where experimental observation is terminated)

$Z = e_n - s$

$R = r - s$

These two cases represent the outcome of two competing processes—the disappearance of the transient and the reconfiguration process of the operating system. In the first case, Z is a random variable which represents the duration of transient errors given that reconfiguration does not occur and R is a random variable which represents the reconfiguration time given that reconfiguration occurs. Since the operating system does not record error detections after the reconfiguration process, the time of disappearance of transient errors can only be observed when reconfiguration does not occur. Similarly, no information is available about the reconfiguration process when the errors disappear first and no reconfiguration takes place. Thus, although one can postulate the existence of some theoretical underlying competing distributions, say $F_{R^*}(r)$ and $F_{Z^*}(z)$, only the conditional distributions

$$\begin{aligned} F_R(r) &= \text{Prob}[\ R < r\ ] \\ &= \text{Prob}[\ R^* < r \mid R^* < \infty\ ] \end{aligned}$$

$$\begin{aligned} F_Z(z) &= \text{Prob}[\ Z < z\ ] \\ &= \text{Prob}[\ Z^* < z \mid R^* = \infty\ ] \end{aligned}$$

can be directly observed. Furthermore, it has been shown that it is impossible to identify the underlying distributions given the conditional distributions and that although the actual underlying distributions may not be independent, the stochastic behavior can be accurately modeled as competing independent processes. (See ref. 3.) Therefore, the use of a semi-Markov model to describe this phenomena is justified. The notation $R^* = \infty$ indicates the case where reconfiguration does not occur. If no errors are detected in an injection, Z is defined to be zero. The results of each injection can only be observed for a finite time. The censoring point C of this preliminary experiment was two minutes. Consequently, the effects of a fault with extremely long latency periods would be missed.

The following model describes the response of the operating system to transient faults with exponential arrival rate λ_T:

$$\begin{tikzcd}
0 \arrow[r, "\lambda_T"] & 1 \arrow[r, "F_R\ (r)"] \arrow[l, bend left, "F_Z\ (z)"] & 2
\end{tikzcd}$$

Since we are dealing with experimental data that is conditional, the nonexponential transitions will be labeled with the conditional distributions. In the above model, the transition from (0) to (1) is the arrival of a transient fault with exponential rate λ_T. The transition from (1) to (0) is the disappearance of the transient errors. Given that this transition occurs, the total elapsed time of the transition is a sample from the distribution $F_Z(z)$. The transition from (1) to (2) is the removal of the faulty processor via reconfiguration. The distribution of reconfiguration time is $F_R(r)$. The probability $p_R = \text{Prob}[\ R^* < \infty\ ]$ is the probability that the transition (1) to (2) occurs. (The probability that the transition (1) to (0) occurs is $1 - p_R$.)

As mentioned earlier, each transient fault injection is performed by physically holding a fault active for a predetermined duration W. Since this can only be done for a finite set of predetermined durations, we can only observe the random variables R and Z in response to particular transient fault durations $(w_1, w_2, w_3, \ldots w_k)$. Thus, we actually observe samples from the conditional distributions

$F_{R\|W}(t \mid w_i)$	(i.e. distribution of recovery time given that the transient fault is active for duration w_i)
$F_{Z\|W}(t \mid w_i)$	(i.e. distribution of the time of disappearance of errors given that the transient fault is active for duration w_i)

The probability

$$p_R(w) = \text{Prob} [R^* < \infty \mid W = w]$$

corresponds to the fraction of times the system reconfigures in the presence of a transient fault of duration w.

Mathematically,

$$F_R(r) = \int_0^\infty F_{R|W}(t|w)\, dF_W(w)$$

$$F_Z(z) = \int_0^\infty F_{Z|W}(t|w)\, dF_W(w)$$

$$p_R = \int_0^\infty p_R(w)\, dF_W(w)$$

where $F_W(w)$ is the distribution of transient fault durations. The motivation for performing the experiment in this manner is that the distribution of transient fault durations $F_W(w)$ is unknown. If experimental data were available for $F_W(w)$, then the transient fault durations could be sampled randomly and Z and R could be measured directly. This indirect method enables us to construct $F_Z(z)$ and $F_R(r)$ under various assumptions about $F_W(w)$.

FAULT INJECTION METHOD AND DATA CAPTURE

It is impossible to perform transient fault injections at every pin in a processor for all possible transient fault durations. Thus, the fault injection locations were chosen randomly weighted according to the chip failure rates and a small set of transient-fault durations (to be injected at every randomly-selected pin) were predetermined. The chip failure rates were determined using MIL-STD-217D. A list of the failure rates used for the chips in the SIFT processors are provided in Appendix A. The set of fault durations were not chosen to be equally far apart (i.e. equal successive differences). This is impractical since the fault latency (i.e. time from injection until first error detection) is several orders of magnitude longer for some pins than for others. A spacing appropriate for one pin location in the processor would not be appropriate for another. Consequently, the natural logarithm of the fault durations were chosen to be equally far apart. The following injection durations were used:

1 μs, 3.16 μs, 10 μs, 31.62 μs, 100 μs, 316.22 μs, 1 ms, 3.162 ms, 10 ms, 31.62 ms, 100 ms, 316.22 ms, 1 s.

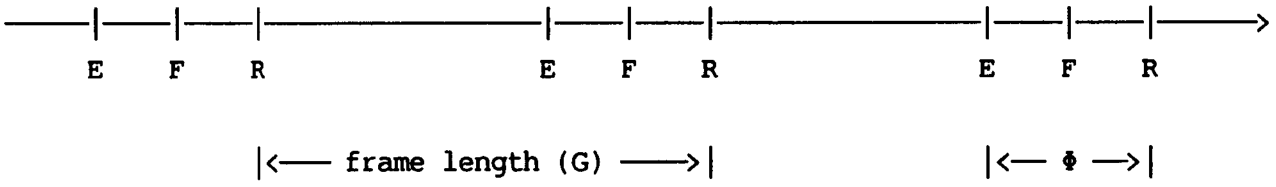
The SIFT operating system was instrumented to obtain the time of each error detection on the non-injected processors. This time was obtained on each processor from a global clock with millisecond resolution. Since error detection is accomplished by voting, error detection is possible only in subframes where voting occurs. The SIFT schedule table including the number of variables voted per subframe is shown below:

subframe	clock tic	task	# variables voted
1	0	CLKTA	
2	2	ICT1	
3	6	ICT2	3
4	9	ICT3	
5	14	MLS	1
6	16	GUIDA	3
7	18	PITCH	6
8	20	LATER	4
9	22	ERRTA	2
10	24	NULLT	
11	26	ICT1	
12	30	ICT2	3
13	33	ICT3	
14	38	MLS	1
15	40	GUIDA	3
16	42	PITCH	6
17	44	LATER	4
18	46	FAULT	2
19	49	NULLT	2
20	51	ICT1	

21	55	ICT2	3
22	58	ICT3	
23	63	MLS	1
24	65	GUIDA	3
25	67	PITCH	6
26	69	LATER	4
27	71	RECFT	2

The SIFT scheduler repeatedly executes the sequence of tasks enumerated in the table above. Each execution of these tasks constitutes a global frame.

The error task ERRTA counts the number of vote errors since its last execution. If this count exceeds the value of parameter THRESHOLD (arbitrarily set to 3) then it sets an error flag ERR[p] indicating that it has diagnosed processor p as faulty during this global frame. The fault-isolation task FAULT retrieves a voted version of ERR[p]. If ERR[p] is true for a processor p for K (arbitrarily set to 2) consecutive global frames then the fault-isolation task tells the reconfiguration task RECFT to remove processor p. In the following diagram, E represents the ERRTA task, F represents the fault-isolation task FAULT and R represents the reconfiguration task RECFT:



If a fault generates errors at a rate greater than THRESHOLD/G then the reconfiguration time will vary between $(K-1)G+\Phi$ and $KG+\Phi$. The value of Φ could be reduced by moving the ERRTA and FAULT task immediately before the RECFT task. This would reduce the mean reconfiguration time in SIFT. The fol lowing illustrates a typical sequence of errors detected by the SIFT processors when a fault is injected on processor 1:

--------------------- GLOBAL FRAME 27 ---------------------

SF	P1	P3	P4	P5	P6
7	---	5(3)	5(3)	5(3)	5(3)
8	---	8(2)	8(2)	8(2)	8(2)
15	---	37(2)	37(2)	37(2)	37(2)
16	---	40(3)	40(3)	40(3)	40(3)
17	---	43(2)	43(2)	43(2)	43(2)
18	---	48(1)	48(1)	48(1)	48(1)
24	---	74(2)	74(2)	74(2)	74(2)
25	---	77(3)	77(3)	77(3)	77(3)
26	---	80(2)	80(2)	80(2)	80(2)

--------------------- GLOBAL FRAME 28 ---------------------

SF	P1	P3	P4	P5	P6
6	---	109(2)	109(2)	109(2)	109(2)
7	---	112(3)	112(3)	112(3)	112(3)
8	---	115(2)	115(2)	115(2)	115(2)
16	---	147(3)	147(3)	147(3)	147(3)
17	---	150(2)	150(2)	150(2)	150(2)
18	---	155(2)	155(2)	155(2)	155(2)
24	---	181(2)	181(2)	181(2)	181(2)
26	---	187(2)!	187(2)!	187(2)!	187(2)!
RECONFIGURATION		187	187	187	187

The first column is the subframe; the remaining columns contain the error-detection times observed on each processor. For example, the column of numbers under the header P4 contains the times that processor P4 detected faults on processor P1. The number in parentheses following each error-detection time is the number of vote errors at that time.

A summary of the results of the 297 transient fault injections is given in Appendix B.

TRANSIENT FAULT CLASSIFICATION

The errors produced by the injected transient faults fell into the following classes:

Transient Null - the injected fault produced no errors

Transient Benign - the injected fault produced a finite sequence of errors, followed by correct operation of the processor

Transient Persistent - the injected fault produced a non-terminating sequence of errors (until reconfiguration).

The transient persistent fault's behavior is indistinguishable from a permanent fault while the system is operating. However, when the injected processor is manually restarted, it operates properly. One way that a transient fault can disable a processor is by crashing the microcode. Although the physical cause of the fault is temporary, the effect is permanent. Transient persistent faults should be diagnosed as permanent by the operating system and removed.

Because the error generation process cannot be observed indefinitely, it is impossible to exactly differentiate between the benign and persistent class. Furthermore, since the SIFT operating system reconfigures in the presence of errors (terminating the error observation process), the problem of distinguishing between the two classes is further complicated. In the section entitled "Future Experimental Directions" a new experimental approach to this problem is described.

In the following summary tables, the following assumptions were made:

(1) If the system reconfigured and errors persisted up to the reconfiguration point, it is assumed that the operating system properly diagnosed the fault as persistent.

(2) If the system reconfigured, but errors disappeared at least 5 ms prior to the reconfiguration, it is assumed that the operating system improperly diagnosed a benign fault as persistent.

(3) If the fault produced errors but the system did not reconfigure, then it is assumed that the fault was benign.

(4) If a fault had not generated an error within the censoring time of the experiment (i.e., 1 minute), the fault was null.

CPU

W (μsec)	Null	Benign	Persistent
0 - 10	.75	.07	.18
10 - 100	.36	.15	.49
100 - 1000	.31	.15	.54
1000 - 10000	.00	.11	.89
10000 - 100000	.00	.21	.79
100000 - 1000000	.00	.05	.95

Memory

W (μsec)	Null	Benign	Persistent
0 - 10	.60	.03	.37
10 - 100	.30	.10	.60
100 - 1000	.17	.11	.72
1000 - 10000	.00	.00	1.00

RELIABILITY ANALYSIS OF SIFT

In this section a methodology for performing a reliability analysis of the SIFT computer system subject to transient and permanent faults will be presented. The methodology will be illustrated by application to a 4-processor SIFT system. The following assumptions govern this model:

1. The system initially consists of four statistically-independent processors which fail permanently at constant failure rate λ_p and transiently at constant rate λ_T.

2. Each processor executes the exact same program on exactly the same inputs so that all non-faulty processors produce exactly the same output. The system "votes" the outputs prior to external use. Thus, so long as a majority of the processors are non-faulty, any erroneous values are "masked".

3. The system removes the faulty processors via reconfiguration. The first reconfiguration reduces the system to a triplex configuration. A second reconfiguration reduces the system to a simplex.

4. The distribution of reconfiguration time $F_R(r)$ is unknown and must be determined experimentally.

5. The distribution of transient error duration $F_Z(z)$ and transient fault duration $F_W(w)$ is unknown. No experimental data is available (nor does this experiment provide any) for these distributions.

The computation of the probability of system failure based on this model will be performed using the Semi-Markov Unreliability Range Evaluator (SURE) program. (See ref. 4.) A key advantage of the SURE program lies in its use of means and variances of the unknown distributions. It is not necessary to assume some family of underlying distribution and perform distribution-fitting procedures. The SURE input file describing this model is:

```
LAMBDA = 2E-4;                    (* permanent fault arrival rate λ_P       *)
GAMMA = 10*LAMBDA;                (* transient fault arrival rate λ_T       *)

P_R =                             (* probability of reconfiguration p_R     *)
MU_R =                            (* mean reconfiguration time μ(F_R)       *)
SIGMA_R =                         (* stan. dev. of reconf. time σ(F_R)      *)
MU_Z =                            (* mean last error time μ(F_Z)            *)
SIGMA_Z =                         (* stan. dev. of last error σ(F_Z)        *)

MU_P =                            (* mean permanent reconf. time μ(F_p)     *)
SIGMA_P =                         (* stan. dev. perm. reconf. time σ(F_p)   *)

1,2 = 4*GAMMA;
2,3 = 3*GAMMA + 3*LAMBDA;
1,4 = 4*LAMBDA;
4,5 = 3*GAMMA;
2,5 = 3*LAMBDA;
4,6 = 3*LAMBDA + 3*GAMMA;
2,7 = <MU_R, SIGMA_R, P_R>;
2,1 = <MU_Z, SIGMA_Z, 1-P_R>;
4,7 = <MU_S, SIGMA_S>;
7,8 = 3*GAMMA;
8,9 = 2*GAMMA + 2*LAMBDA;
7,10  = 3*LAMBDA;
10,12 = 2*LAMBDA + 2*GAMMA;
10,11 = 2*GAMMA;
8,12  = 2*LAMBDA;
8,13 = <MU_R, SIGMA_R, P_R>;
10,13 = <MU_S, SIGMA_S>;
8,7 =  <MU_Z, SIGMA_Z, 1-P_R>;
13,14 = GAMMA + LAMBDA;
```

The graphical display of this model in figure 1 was generated by the SURE program. The complete input file to the SURE program including the calculation of the means and variances is given in Appendix C.

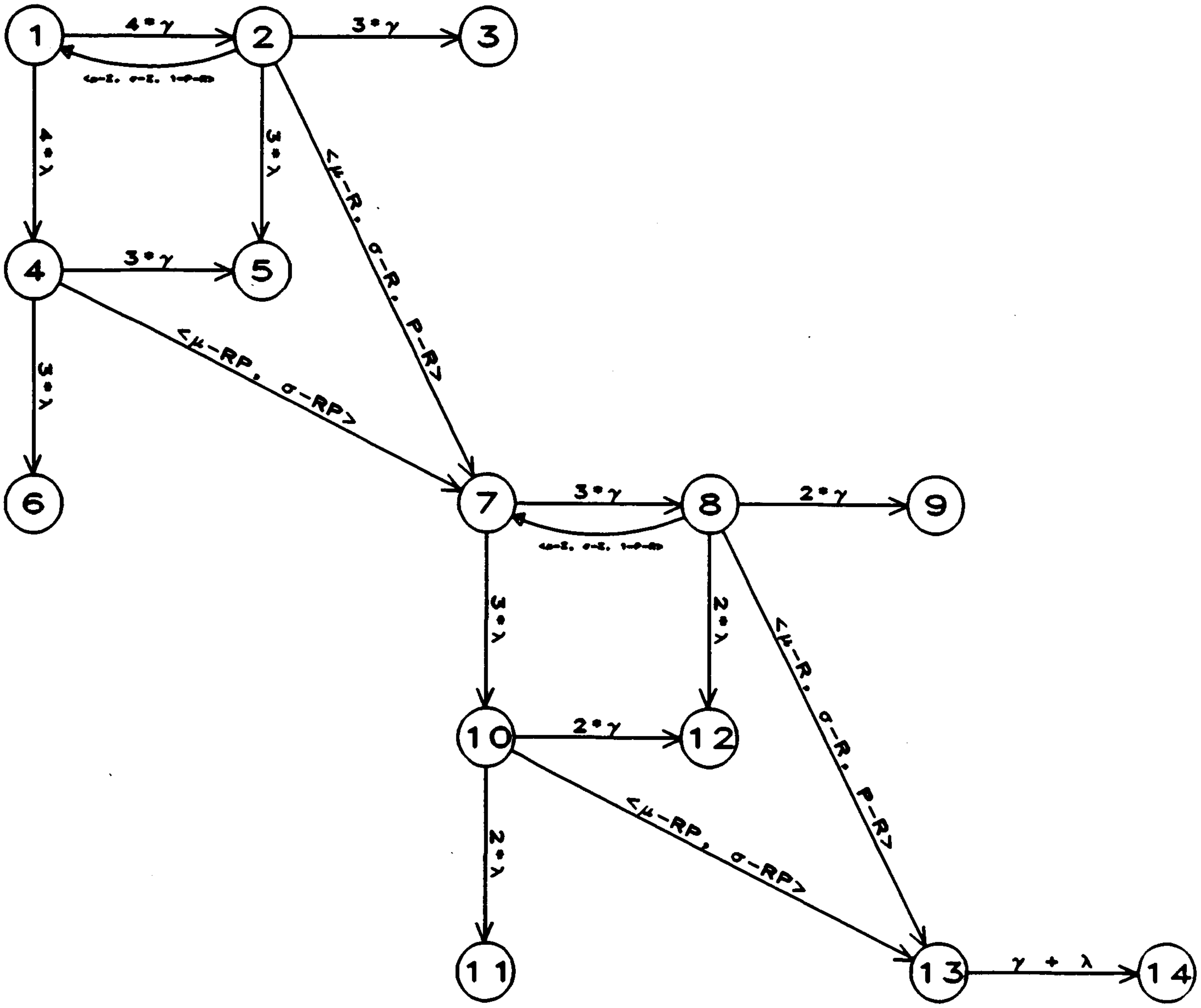


Figure 1.- Reliability model of 4-processor SIFT.

The following non-parametric statistics must be estimated from the experimental data:

$$p_R = \int_0^\infty p_R(w)\ dF_W(w)$$

$$\mu(F_R) = \int_0^\infty E[R|W = w]\ dF_W(w)$$

$$\sigma^2(F_R) = E(R^2) - (E[R])^2 = \int_0^\infty E[R^2|W = w]\ dF_W(w) - [\mu(R)]^2$$

$$\mu(F_Z) = \int_0^\infty E[Z|W = w]\ dF_W(w)$$

$$\sigma^2(F_Z) = E(Z^2) - (E[Z])^2 = \int_0^\infty E[Z^2|W = w]\ dF_W(w) - [\mu(Z)]^2$$

$$\mu(F_p) = E[R|\ W = \infty]$$

$$\sigma(F_p) = E[R^2|W = \infty]$$

Since we only have measurements of $E[R|W{=}w_i]$, $E[Z|W{=}w_i)$, $E[R^2|W{=}w_i]$ and $E[Z^2|W{=}w_i)$ for a few values of w_i we are forced to approximate the integral with a numerical method:

$$p_R = \int_0^\infty p_R(w)\ dF_W(w)$$

$$= \sum_{i=1}^{k+1} \int_{w_{i-1}}^{w_i} p_R(w)\ dF_W(w) \qquad \text{where } w_{k+1} = \infty$$

$$\simeq \sum_{i=1}^{k} \hat{p}_R(w_i)\ [F_W(w_i) - F_W(w_{i-1})] = \hat{p}_R$$

Similarly

$$\hat{\mu}(F_R) \simeq \sum_{i=1}^{k} \hat{E}[R|W=w_i] \, [F_W(w_i) - F_W(w_{i-1})]$$

$$\hat{\sigma}^2(F_R) \simeq \sum_{i=1}^{k} \hat{E}[R^2|W=w_i] \, [F_W(w_i) - F_W(w_{i-1})] - \mu^2(F_R)$$

$$\hat{p}_Z = 1 - \hat{p}_R$$

$$\hat{\mu}(F_Z) \simeq \sum_{i=1}^{k} \hat{E}[Z|W=w_i] \, [F_W(w_i) - F_W(w_{i-1})]$$

$$\hat{\sigma}^2(F_Z) \simeq \sum_{i=1}^{k} \hat{E}[Z^2|W=w_i] \, [F_W(w_i) - F_W(w_{i-1})] - \mu^2(F_Z)$$

The first and second moments of the distribution of reconfiguration time in the presence of permanent faults F_P can be estimated using the following simple unbiased estimators:

$$\hat{\mu}(F_P) = \sum_{i=1}^{\eta} r_i/\eta$$

$$\hat{\sigma}^2(F_P) = \sum_{i=1}^{\eta}(r_i - \hat{\mu})^2/(\eta-1)$$

where $(r_1, r_2, \ldots r_\eta)$ is a random sample obtained via permanent fault injection. A histogram of this sample is given in figure 2. The mean reconfiguration time is 272.6 ms and the standard deviation is 121.5 ms.

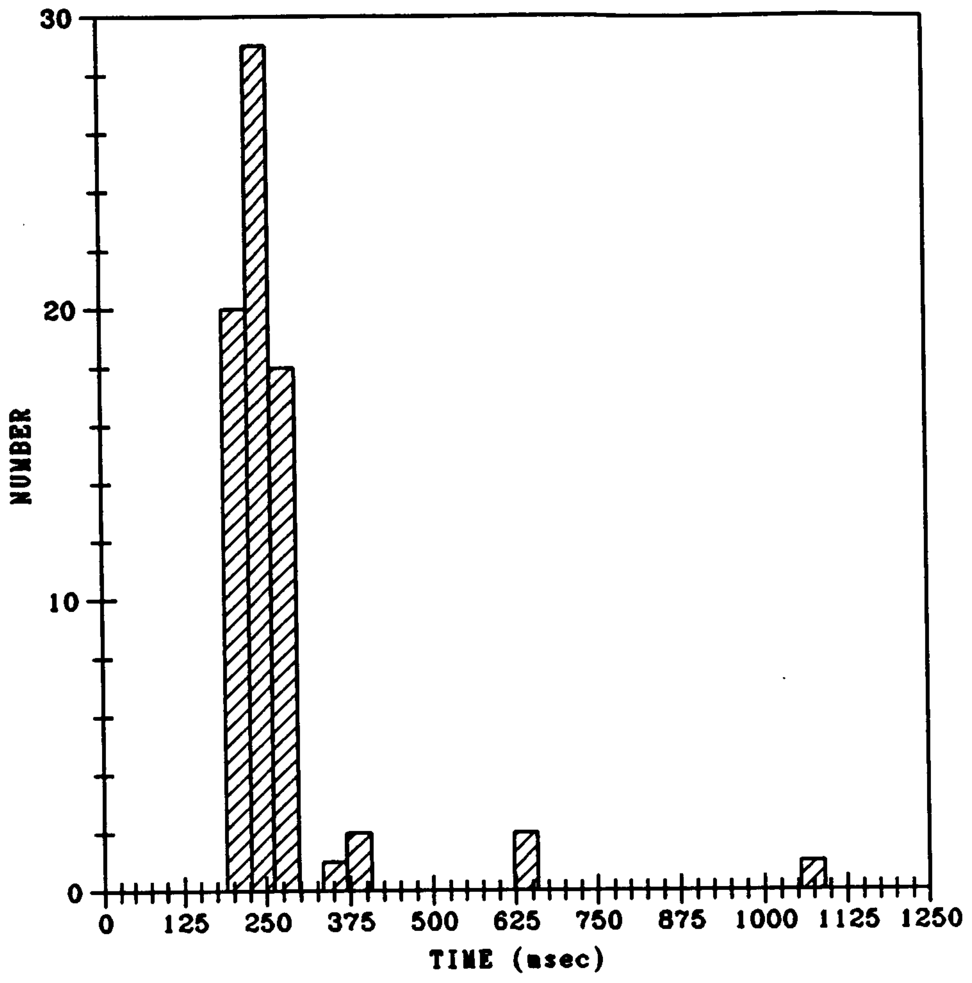


Figure 2.- Reconfiguration time histogram (permanent faults).

The following values of $\hat{p}_R$, $\hat{E}[R|W=w_i]$, $\hat{E}[Z|W=w_i]$, $\hat{E}[R^2|W=w_i]$ and $\hat{E}[Z^2|W=w_i]$ were observed:

w_i	$\hat{p}_R(w)$	$\hat{E}[Z\|W=w_i]$	$\hat{E}[R\|W=w_i]$	$\hat{E}[Z^2\|W=w_i]$	$\hat{E}[R^2\|W=w_i]$
1 μs	.17	0.1	239.0	0.16	57282.60
3.16 μs	.30	66.3	350.9	92402.34	181682.00
10 μs	.37	0.0	239.4	0.00	58599.27
31.62 μs	.53	47.3	255.9	15875.00	68679.12
100 μs	.69	0.0	247.1	0.00	62566.75
316.22 μs	.54	2.5	238.3	40.00	57448.20
1 ms	.86	0.0	284.3	0.00	108444.48
3.162 ms	1.00	—	256.9	—	66597.96
10 ms	1.00	—	249.1	—	62921.78
31.62 ms	.95	282.0	248.4	79524.00	62708.11
100 ms	.90	99.0	255.7	9801.00	66016.66
316.22 ms	1.00	—	251.5	—	64170.50
1 s	1.00	—	236.7	—	56681.30

Since the distribution of transient fault durations is unknown and no experimental data is available, sensitivity analysis will be performed under the assumption of three different families of distributions. If experimental data were available for $F_W(w)$, then the transient fault durations could be sampled randomly and Z and R could be measured directly. In that case, this indirect calculation would be unnecessary. The exponential, uniform and Weibull distributions will be analyzed.

Analysis Assuming Exponential Transient Duration

In this section the reliability analysis is performed under the assumption that the distribution of the duration of transient faults is exponentially distributed. Thus,

$$F_W(w) = 1 - e^{-\phi w}$$

for some ϕ. The probability of system failure as a function of $\mu(F_W) = 1/\phi$ is given in figure 3.

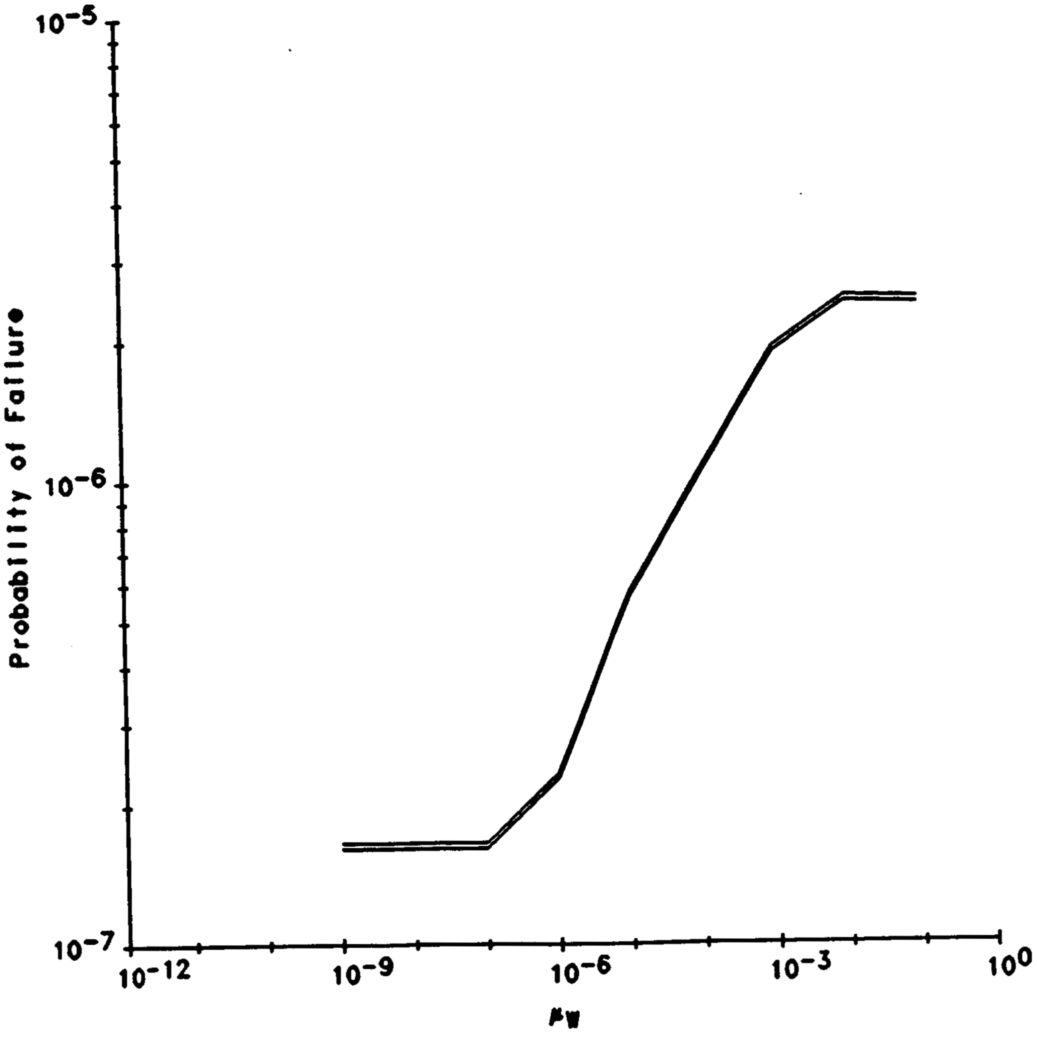


Figure 3.- Prob. of failure vs. $\mu_w(F)$ for exponential F.

Analysis Assuming Uniform Transient Duration

In this section the reliability analysis is performed under the assumption that the distribution of the duration of transient faults is uniformly distributed. Thus,

$$F_w(w) = \begin{cases} w/\beta & 0 \leq w \leq \beta \\ 1 & \beta \leq w \end{cases}$$

for some β. The probability of system failure as a function of $\mu(F_w) = \beta/2$ is given in figure 4.

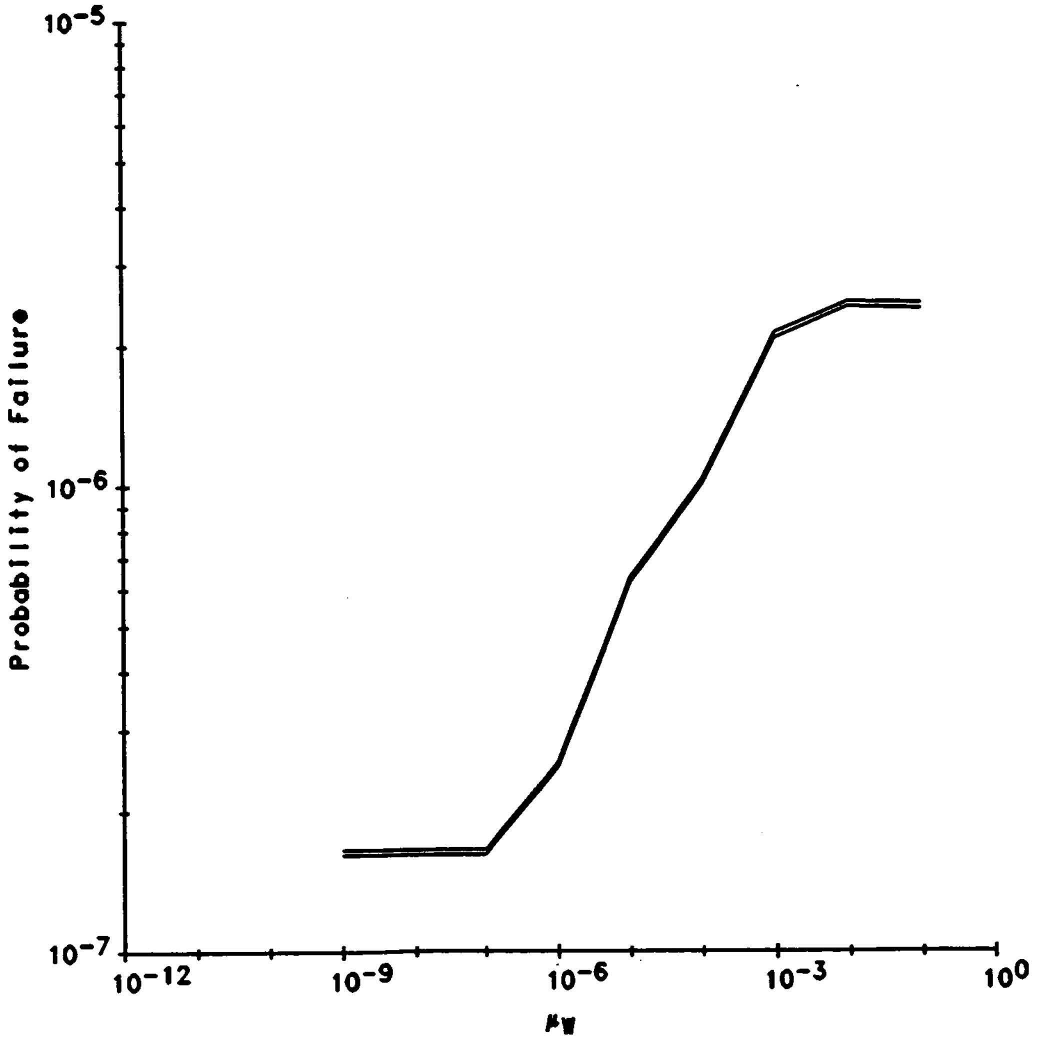


Figure 4.- Prob. of failure vs. $\mu_w(F)$ for uniform F.

Analysis Assuming Weibull Transient Duration

In this section the reliability analysis is performed under the assumption that the distribution of the duration of transient faults is Weibull:

$$F_w(w) = 1 - e^{-\phi w^{\alpha}}$$

for some ϕ and α. The probability of system failure as a function of $\mu(F_w) = (1/\phi)^{1/\alpha}\,\Gamma(1+1/\alpha)$ for $\alpha=2$ and $\alpha=1/2$ is given in figure 5.

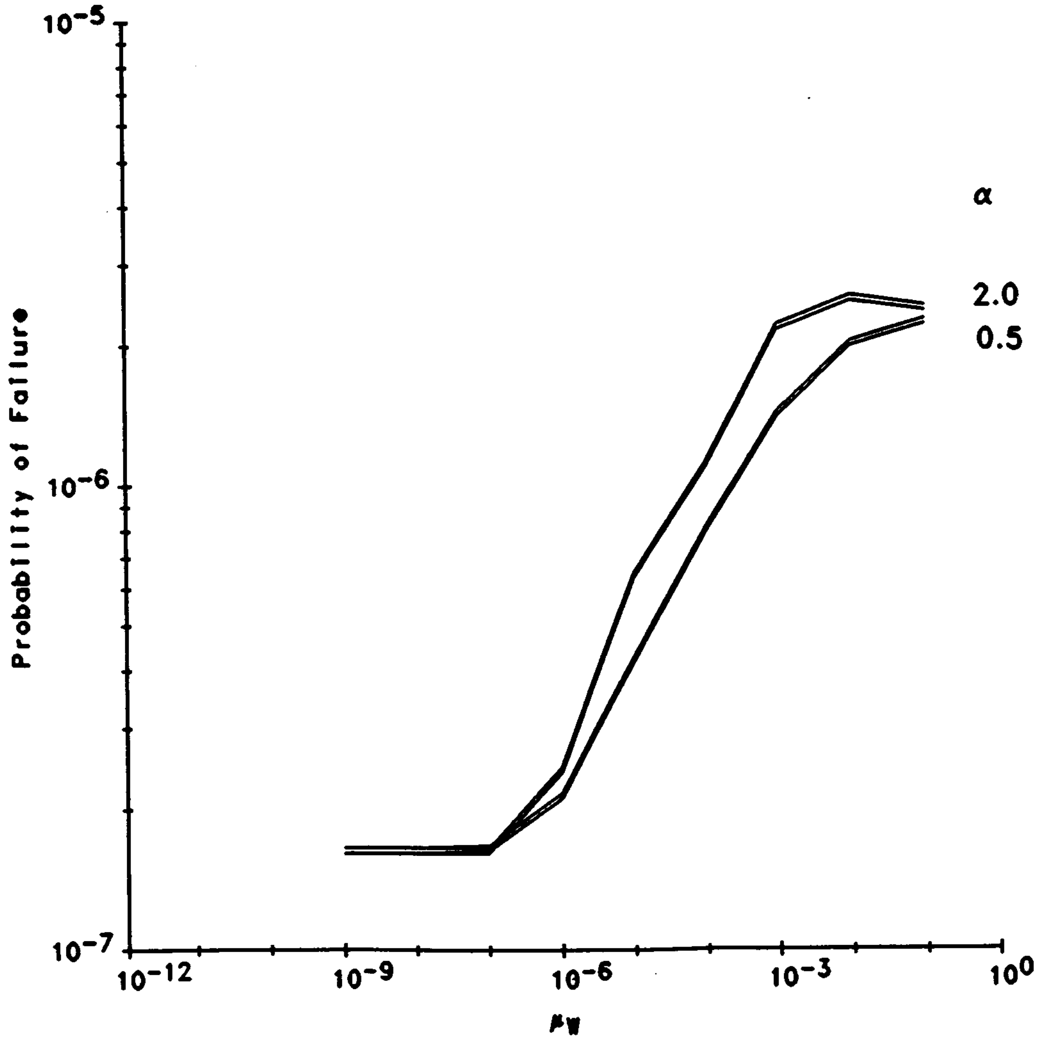


Figure 5.- Prob. of failure vs. $\mu_w(F)$ for Weibull F.

Sensitivity to $F_w(w)$

A comparison of figures 3, 4 and 5 reveals that the probability of system failure is only moderately sensitive to the different shapes of the distributions. However, the unreliability varies over two orders of magnitude depending upon the mean of $\mu(F_w)$. Once again the reader is cautioned that this observation is based on a very small sample and the assumption that F_w comes from one of these three families of distributions.

EFFECTIVENESS OF SIFT'S TRANSIENT/PERMANENT FAULT DISCRIMINATOR

In this section some preliminary observations are made about the effectiveness of the SIFT transient/permanent fault discrimination algorithm. There are two ways the operating system can incorrectly diagnose a fault:

(1) A permanent fault generates intermittent errors that are indistinguishable from the errors produced by two or more transients faults and thus is not reconfigured.

(2) The time that the operating system waits to see if a fault is transient is not long enough to recognize a particularly long transient and consequently reconfigures before the fault disappears.

The first case was not observed during the experiment (i.e., all permanent faults were successfully reconfigured). There were many cases where the transient injection resulted in the injected processor being reconfigured out of the system. However, whether this was a correct decision depends on whether the fault was transient benign or transient persistent. If the fault is transient persistent, the decision was correct. If the fault was transient benign, the decision was incorrect. Because the error generation process was not observed after a reconfiguration, the type of fault could not be ascertained with 100% confidence from the experimental data. Thus, it was impossible to determine if at some time subsequent to the reconfiguration the errors of a transient benign fault would disappear. Also, if the errors had disappeared shortly before the reconfiguration (suggesting that the fault was transient benign), there was no way to determine if this was merely a temporary lapse in the error sequence of a transient persistent fault. In the section entitled "FUTURE EXPERIMENTAL DIRECTIONS" a modification to the experimental approach is presented which facilitates the process of distinguishing transient benign faults from transient persistent faults. In the rest of the section a simple analysis of the data is given which gives some indication of the operating system's ability to distinguish these types of faults.

Let S be the time between the last error detection and the reconfiguration. If a fault generates errors up to the time of reconfiguration (i.e. $S \approx 0$), then the diagnosis as permanent is probably correct. However, if S is large, then most likely the fault was improperly diagnosed. The distribution of S is given in figure 6.

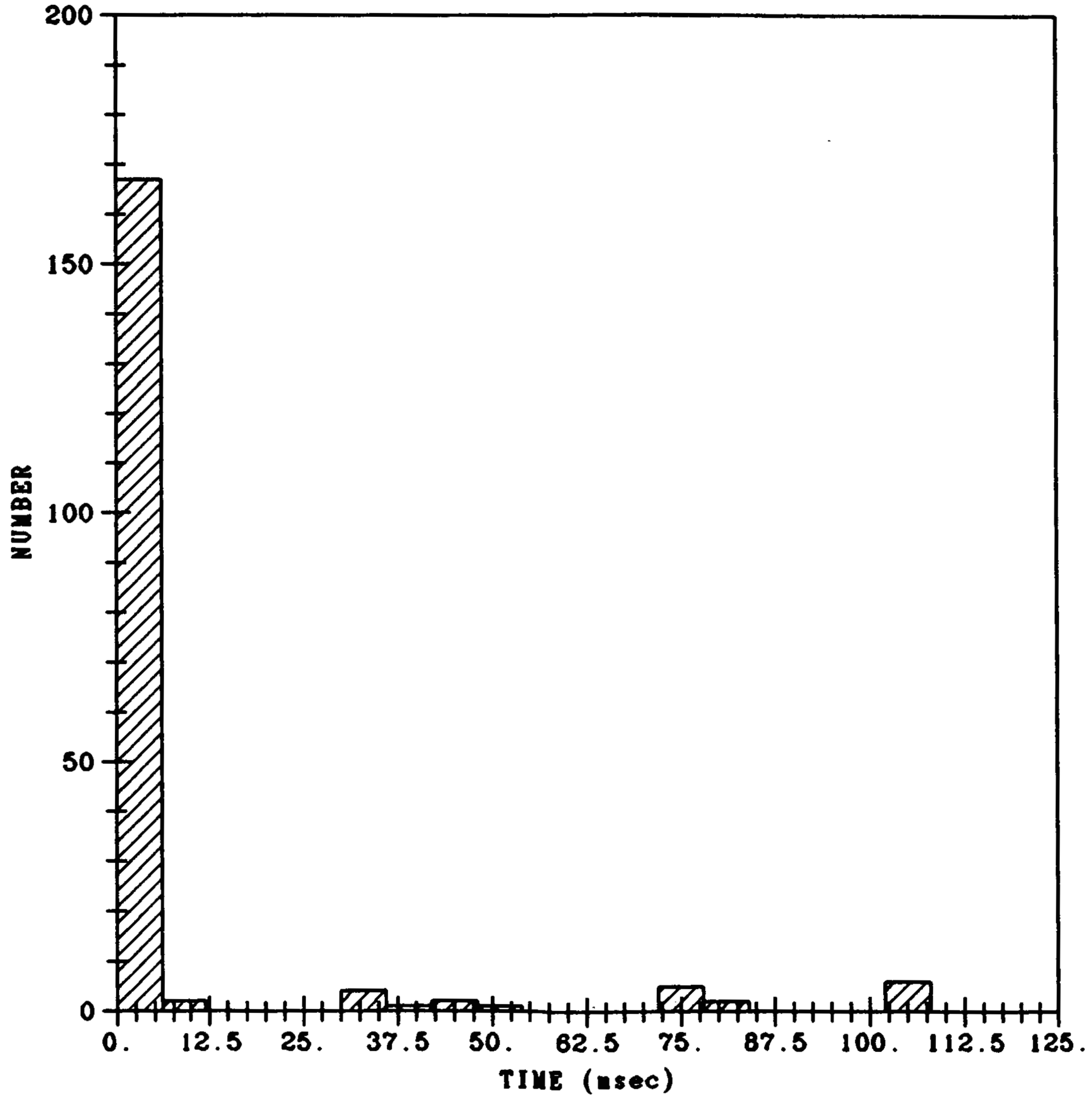


Figure 6.- Histogram of $S = R - Z^*$.

There were 190 fault injections which resulted in a reconfiguration. Of these, 167 had a value of S less than one clock tick (1.6 ms). In the remaining injections there were only 2 injections with a value of S less than 31 ms as revealed in the following table:

S	# of occurrences
0 - 1 ms	167
2 - 5 ms	0
6 ms	2
7 - 31 ms	0
> 31 ms	21

Exactly where the division should be made between the transient benign and transient persistent class is not obvious. Since there were very few injections with values of S between 2 ms and 31 ms, any choice in this range would yield essentially the same results. In the following tables, the division is made at 5 ms. Therefore all faults with an value of S less than 5 is assumed to be transient persistent and those with a value of S greater than 5 are assumed to be transient benign. Using this classification scheme, the percentage of improperly reconfigured faults were:

$$\% \text{ error} = \frac{\#\text{ transient benign faults reconfigured}}{\#\text{ reconfigurations}} \times 100\%$$

$$= \frac{23}{190} \times 100\% = 12.1\%$$

There were 9 injections which produced errors but did not lead to a reconfiguration. Assuming that these faults were correctly diagnosed as transient benign, the percentage of transient benign faults which were improperly diagnosed were:

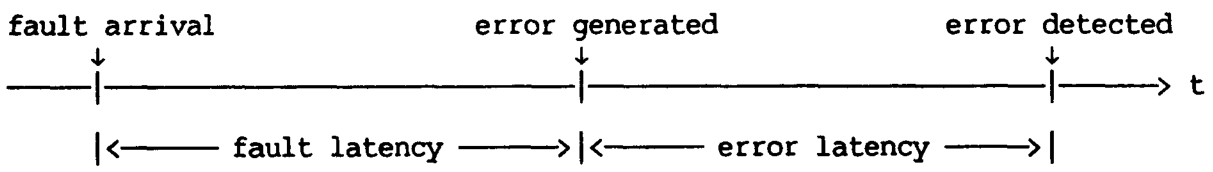
$$\% \text{ error} = \frac{\#\text{ transient benign faults reconfigured}}{\#\text{ transient benign faults}} \times 100\%$$

$$= \frac{23}{32} \times 100\% = 71.9\%$$

Since most of the faults injected were transient persistent the percentage of improperly reconfigured faults was small (12.1%). However, of the faults that should not have been reconfigured (transient benign), 71.9% were improperly reconfigured.

DISTRIBUTION OF FAULT LATENCY

The data obtained in this experiment is sufficient to determine fault latency in SIFT using the methodology developed by the University of Michigan. (See ref. 5.) Consider the following graph of the fault propagation process:



Let L be a random variable representing the fault latency with distribution function $F_L(l)$ and let $n_i(w)$ represent the total number of transient fault injections at pin i with duration W. If $D_i(w)$ is a random variable representing the number of injections which result in at least one error detection, then

$$E[\; D_i(w)/n_i(w) \;] \leq F_L(w)$$

Under the assumption that errors generated by the injected fault are propagated and detected before the censoring point of the experiment:

$$E[\; D_i(w)/n_i(w) \;] = F_L(w)$$

If $d_i(w)$ detections are observed in response to $n_i(w)$ injections, the following estimator of $F_L(w)$ is unbiased:

$$\hat{F}_L(w) = d_i(w)/n_i(w) \simeq F_L(w).$$

The following measurements were obtained:

w_i	$d_i(w_i)$	$n_i(w_i)$	$\hat{F}_L(w_i)$
1 μs	6	30	.20
3.16 μs	10	30	.33
10 μs	11	30	.37
31.62 μs	19	30	.63
100 μs	20	29	.69
316.22 μs	17	28	.61
1 ms	25	29	.86
3.162 ms	24	24	1.00
10 ms	18	18	1.00
31.62 ms	19	19	1.00
100 ms	10	10	1.00
316.22 ms	10	10	1.00
1 s	10	10	1.00

Although $F_L(w)$ must be monotonic increasing, the estimates $\hat{F}_L(w_i)$ may not be. Clearly, a statistical method of estimating the $F_L(w_i)$ under a constraint of monotonicity is needed.

Theoretically:

$$\mu(F_L) = \int_0^\infty 1 - F_L(t)\, dt$$

$$\sigma^2(F_L) = 2 \int_0^\infty t\, [1 - F_L(t)]\, dt \;-\; [\mu(F_L)]^2$$

The following are approximations to the mean and variance:

$$\hat{\mu}(F_L) = \sum_{i=1}^{k} [1 - \hat{F}_L(w_i)]\, (w_i - w_{i-1})$$

$$\hat{\sigma}^2(F_L) = \sum_{i=1}^{k} 2w_i\, [1 - \hat{F}_L(w_i)]\, (w_i - w_{i-1}) - [\hat{\mu}(F_L)]^2$$

The following values of $\hat{\mu}$ and $\hat{\sigma}$ were obtained:

$\hat{\mu} = 0.216$ ms

$\hat{\sigma} = 0.451$ ms

FUTURE EXPERIMENTAL DIRECTIONS

Improvements in Measuring the Effectiveness of the Operating System's Transient/Permanent Fault Discriminator

A key factor in evaluating the effectiveness of the operating system's ability to discriminate between transient and permanent faults is determining whether an injected transient fault is transient benign or transient persistent. In this section a simple modification to the experimental approach is described which enables a more accurate determination of the type of the fault.

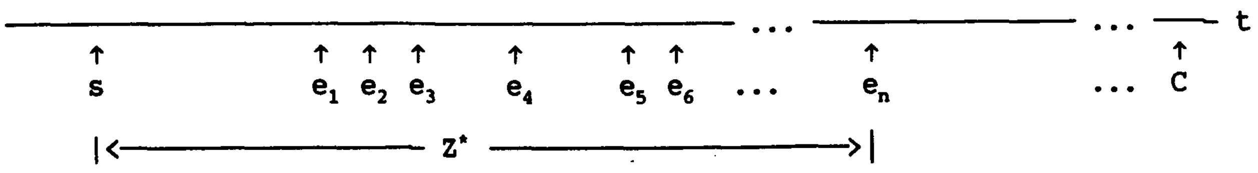
The recommended change in the experimental method is:

(1) disable the reconfiguration process of the SIFT operating system so that reconfiguration does not occur.

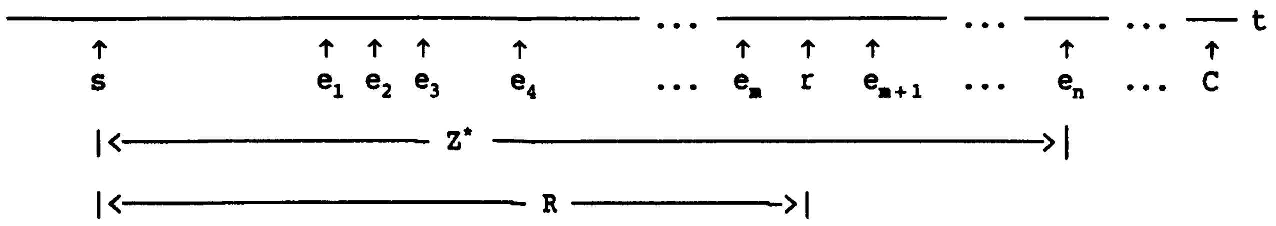
(2) instrument the operating system to record the time that reconfiguration would normally have occurred.

In this way the error propogation process can be observed for a greater amount of time. The possible results of an injection are now:

case 1: no reconfiguration



case 2: reconfiguration occurs


t
s
e_1 e_2 e_3 e_4 ... e_m r e_{m+1} ... e_n ... C
Z^*
R

where

s = time injection begins

e_i = the time of detection of the ith error $(1 \le i \le n)$

r = time operating system reconfigures

C = censoring point of the experiment

$Z^* = e_n - s$

$R = r - s$

By observing the error generation process after the reconfiguration time r until the censoring point C, we obtain the unconditional Z^* directly. Therefore, the incorrect diagnosis of a transient fault as a permanent can be more accurately discerned. If the errors disappear at some point after the reconfiguration point, then the diagnosis that the fault was permanent was wrong. Similarly, the classification of the transient faults into transient-benign and transient-persistent is simplified.

Measuring $F_z(z)$ and the Consequent Refined Analysis Method

In this section, a method of measuring the duration of natural (i.e. non-injected) transient errors $F_z(z)$ is introduced. The implications of such an experiment are far-reaching. First, the need to assume a simple stuck-at-1 pin-level fault has been removed. Second, the need for assuming some underlying distribution for the duration of the faults $F_w(w)$ is eliminated. The response of the operating system to transient faults of varying durations created by physical pin-level injections no longer has to be measured. The observed response of the operating system to the natural transient faults can be directly entered into the reliability model. The effectiveness of any modifications to the transient/permanent fault discrimination algorithm can be measured by artificially introducing error detections. These artificial error detections can be introduced by changing memory locations in the SIFT processors while they are executing. The patterns of error detections to be introduced artifically can be inferred from the sequences of error detections observed from natural transient faults.

The following approach is suggested for measurement of $F_Z(z)$. Disengage SIFT's reconfiguration algorithm and let it run continuously for many years. Instrument the operating system with the same data gathering code as described in the section "EXPERIMENTAL APPROACH" and collect as many natural transient faults as possible. (Note. the distribution F_Y can be determined from the distributions F_L and F_Z using the above relationship between the random variables). If the transient fault arrival rate is $5x10^{-3}$/hour, about 40 transient faults should be observed in a year.

ACKNOWLEDGEMENT

The authors are grateful to Quyen Duong Cleary for her help in performing this experiment. Without her patient performance of hundreds of physical fault injections this paper would only be a theoretical discussion.

CONCLUDING REMARKS

A detailed description of the preliminary transient fault experiment along with the results from 297 transient injections are given. Although not enough data was obtained to draw statistically significant conclusions, the foundation has been laid for a large-scale transient fault experiment. Several changes in the experimental procedure are recommended for the large-scale experiment in order to increase the usefulness of the experiment. The sensitivity of the probability of system failure to the mean duration of the transient faults reveals the pressing need for credible measurements of transient fault behavior.

APPENDIX A

SIFT Chip Failure Rates (x 10^{-6} hr^{-1})

Chip Type	# pins	Rate/Chip	Chip Type	# pins	Rate/Chip
54S30	14	0.1654	54S20	14	0.1913
5440	14	0.2132	54S244	20	0.3099
54LS30	12	0.1870	54LS20	12	0.1913
54LS21	12	0.1913	4001B	14	0.2387
4093B	14	0.2388	5410	14	0.2397
54LS11	14	0.2404	54LS10	14	0.2404
54LS27	14	0.2404	54S10	14	0.2406
54126	14	0.2424	5438	14	0.2424
54125	14	0.2424	54LS86	14	0.2432
54LS02	14	0.2432	54LS09	14	0.2432
54LS08	14	0.2432	54LS33	14	0.2432
54LS32	14	0.2432	54LS125	14	0.2432
54LS00	14	0.2432	54LS126	14	0.2432
54S37	14	0.2437	54S86	14	0.2437
54S08	14	0.2437	54S32	14	0.2437
54S02	14	0.2437	54S00	14	0.2437
54LS122	12	0.2100	54LS53	14	0.2456
54155	16	0.2815	5404	14	0.2472
54LS51	14	0.2479	54LS04	14	0.2479
54S04	14	0.2495	54S51	14	0.2495
70C96	16	0.2899	5437	16	0.2916
5474	14	0.2580	54LS74	14	0.2584
54LS74A	14	0.2584	7837	16	0.2964
54S74	14	0.2615	54C175	16	0.3001
54C174	16	0.3010	54LS93	10	0.1883
54LS279	16	0.3012	54LS367	16	0.3012
54LS368	16	0.3012	54LS113	14	0.2643
54LS92	10	0.1895	7835	16	0.3047
DS1651	16	0.3070	54S113	14	0.2702
7603.2	16	0.3098	54S288	16	0.3098
5331	16	0.3098	HM7603	16	0.3098
HD6440A.2	18	0.3484	54LS288	16	0.3119
54LS158	16	0.3121	54LS155	16	0.3121
54LS157	16	0.3121	54LS257	16	0.3121
54156	16	0.3123	54LS138	16	0.3135
54LS153	16	0.3135	54LS253	16	0.3135
54LS112	16	0.3135	54LS109	16	0.3135
54LS352	16	0.3135	54LS151	16	0.3149
54LS251	16	0.3149	54LS139	16	0.3163
AM2902	16	0.3176	2902	16	0.3176
54182	16	0.3189	54LS123	16	0.3189
54S112	16	0.3194	54S153	16	0.3194
54S253	16	0.3194	54S151	16	0.3217
54LS175	16	0.3240	LM119D	14	0.2859
54175	16	0.3271	54LS148	16	0.3301
54LS148	16	0.3301	54LS164	14	0.2891
54LS241	20	0.4129	54LS240	20	0.4129

54LS244	20	0.4129	29LS18	16	0.3313
54S240	20	0.4179	54LS174	16	0.3383
54S175	16	0.3385	DM7136	16	0.3392
5485	16	0.3392	7136	16	0.3392
54LS245	20	0.4242	25LS2518	16	0.3405
7611.2	16	0.3456	54LS393	14	0.3049
54LS194	16	0.3507	5496	16	0.3542
54LS298	16	0.3552	7131	16	0.3583
54LS161A	16	0.3620	54LS161	16	0.3620
25LS2537	20	0.4530	54LS163	16	0.3632
9410	18	0.4093	54LS191	16	0.3643
54LS259	16	0.3643	54LS390	16	0.3654
54LS169	16	0.3654	75107B	14	0.3209
MHQ3467	14	0.3209	54LS290	16	0.3677
54LS165	16	0.3677	54LS273	20	0.4621
54LS377	20	0.4632	54LS374	20	0.4711
25LS2536	20	0.4803	54273	20	0.4859
25LS377	20	0.4873	SE555F	8	0.1953
55471J	8	0.1953	54S471	20	0.4918
AM25LS2569	20	0.4969	54LS381	20	0.4981
75109A	14	0.3516	25LS2517	20	0.5030
CA3039	12	0.3059	54LS299	20	0.5169
75109	14	0.3809	2911	20	0.5504
9407	24	0.6663	7641.2	23	0.6538
54S472	20	0.5771	HM7643	18	0.5216
AM2940DM	28	0.8395	HM6514.2	18	0.6013
2914	40	1.3708	AM2812	28	1.0408
2901A	40	1.4945	93L422	22	0.9420
2901A	40	1.8233	AM2901A	40	1.8233
LM193	8	0.3770	2716	24	1.4030
MK4114.3	18	1.1236	LM741	7	0.4906
LF156H	8	0.7382	QT6T9	4	0.4906
LM120H.5	3	0.4348			

APPENDIX B

Summary of Fault Injections

This section includes a summary of the 279 transient fault injections in tabular form. The information included under each heading is:

INJ -	injection number
RECTM -	the time in milliseconds from the injection until the system reconfigured.
TWIDTH-	the duration of the injection in milliseconds.
FIRSTE-	the time in milliseconds from the injection until the first error detection on another processor. The notation ... indicates that no error was detected.
LASTE -	the time in milliseconds from the injection until the last error detection on another processor.

==>	indicates last error same as reconfiguration time.
...	indicates no errors detected
-x+R	indicates the last error was x milliseconds before reconfiguration

TYP - the type of fault: SA1 = + 5 volts, SA0 = -5 volts.

LOCATION - the processor, board, chip and pin where fault was injected.

INJ	RECTM	TWIDTH(ms)	FIRSTE	LASTE	TYP	LOCATION			
1	244	0.001	20	==>	SA1	P1	CPU	U35	2
2	...	0.001	...	...	SA1	P1	CPU	U35	2
3	...	0.001	...	...	SA1	P1	CPU	U35	2
4	...	0.001	...	...	SA1	P1	CPU	U35	2
5	236	0.001	12	==>	SA1	P1	CPU	U35	2
6	...	0.001	...	...	SA0	P1	CPU	U35	2
7	...	0.001	...	...	SA0	P1	CPU	U35	2
8	...	0.001	...	...	SA0	P1	CPU	U35	2
9	216	0.001	2	==>	SA0	P1	CPU	U35	2
10	...	0.001	...	...	SA0	P1	CPU	U35	2
11	...	0.003	...	...	SA1	P1	CPU	U35	2
12	284	0.003	23	==>	SA1	P1	CPU	U35	2
13	...	0.003	...	...	SA1	P1	CPU	U35	2
14	...	0.003	...	...	SA1	P1	CPU	U35	2
15	574	0.003	35	-40+R	SA1	P1	CPU	U35	2
16	...	0.003	...	...	SA0	P1	CPU	U35	2

17	...	0.003	...	...	SA0	P1	CPU	U35	2
18	...	0.003	...	...	SA0	P1	CPU	U35	2
19	216	0.003	1	==>	SA0	P1	CPU	U35	2
20	...	0.003	...	...	SA0	P1	CPU	U35	2
21	282	0.010	21	-1+R	SA1	P1	CPU	U35	2
22	259	0.010	4	-108+R	SA1	P1	CPU	U35	2
23	193	0.010	5	==>	SA1	P1	CPU	U35	2
24	...	0.010	...	...	SA1	P1	CPU	U35	2
25	214	0.010	25	==>	SA1	P1	CPU	U35	2
26	...	0.010	...	...	SA0	P1	CPU	U35	2
27	...	0.010	...	...	SA0	P1	CPU	U35	2
28	...	0.010	...	...	SA0	P1	CPU	U35	2
29	...	0.010	...	...	SA0	P1	CPU	U35	2
30	...	0.010	...	...	SA0	P1	CPU	U35	2
31	...	0.032	129	279	SA1	P1	CPU	U35	2
32	253	0.032	2	==>	SA1	P1	CPU	U35	2
33	...	0.032	...	...	SA1	P1	CPU	U35	2
34	255	0.032	3	==>	SA1	P1	CPU	U35	2
35	239	0.032	15	-73+R	SA1	P1	CPU	U35	2
36	250	0.032	26	==>	SA0	P1	CPU	U35	2
37	...	0.032	...	...	SA0	P1	CPU	U35	2
38	...	0.032	...	...	SA0	P1	CPU	U35	2
39	274	0.032	13	-1+R	SA0	P1	CPU	U35	2
40	190	0.032	1	-1+R	SA0	P1	CPU	U35	2
41	241	0.100	17	-1+R	SA1	P1	CPU	U35	2
42	221	0.100	104	==>	SA1	P1	CPU	U35	2
44	284	0.100	23	-6+R	SA1	P1	CPU	U35	2
45	237	0.100	13	-1+R	SA1	P1	CPU	U35	2
46	...	0.100	...	...	SA0	P1	CPU	U35	2
47	287	0.100	26	-1+R	SA0	P1	CPU	U35	2
48	246	0.100	21	-1+R	SA0	P1	CPU	U35	2
49	...	0.100	...	...	SA0	P1	CPU	U35	2
50	188	0.100	3	==>	SA0	P1	CPU	U35	2
51	253	0.316	1	-1+R	SA1	P1	CPU	U35	2
52	...	0.316	8	18	SA1	P1	CPU	U35	2
53	...	0.316	...	...	SA1	P1	CPU	U35	2
54	261	0.316	3	-33+R	SA1	P1	CPU	U35	2
55	221	0.316	3	==>	SA1	P1	CPU	U35	2
56	248	0.316	24	-107+R	SA0	P1	CPU	U35	2
57	...	0.316	...	...	SA0	P1	CPU	U35	2
58	231	0.316	114	==>	SA0	P1	CPU	U35	2
60	184	0.316	2	-72+R	SA0	P1	CPU	U35	2
61	215	1.000	26	-1+R	SA1	P1	CPU	U35	2
62	216	1.000	2	==>	SA1	P1	CPU	U35	2
63	...	1.000	...	...	SA1	P1	CPU	U35	2
64	215	1.000	26	-1+R	SA1	P1	CPU	U35	2
65	227	1.000	3	==>	SA1	P1	CPU	U35	2
71	264	3.160	3	==>	SA1	P1	CPU	U35	2
72	286	3.160	25	==>	SA1	P1	CPU	U35	2
73	252	3.160	1	-32+R	SA1	P1	CPU	U35	2
74	244	3.160	20	-1+R	SA1	P1	CPU	U35	2
75	278	3.160	17	==>	SA1	P1	CPU	U35	2
81	280	1.000	19	==>	SA0	P1	CPU	U35	2
82	...	1.000	...	...	SA0	P1	CPU	U35	2
83	262	1.000	4	==>	SA0	P1	CPU	U35	2

84	263	1.000	2	==>	SA0	P1	CPU	U35	2
85	257	1.000	2	-47+R	SA0	P1	CPU	U35	2
86	262	3.162	4	-1+R	SA0	P1	CPU	U35	2
87	253	3.162	2	==>	SA0	P1	CPU	U35	2
88	281	3.162	20	==>	SA0	P1	CPU	U35	2
89	280	3.162	19	-1+R	SA0	P1	CPU	U35	2
90	220	3.162	3	==>	SA0	P1	CPU	U35	2
91	271	10.000	10	==>	SA1	P1	CPU	U35	2
92	219	10.000	2	==>	SA1	P1	CPU	U35	2
93	285	10.000	24	==>	SA1	P1	CPU	U35	2
94	216	10.000	1	==>	SA1	P1	CPU	U35	2
95	247	10.000	23	-1+R	SA1	P1	CPU	U35	2
96	244	10.000	20	-1+R	SA0	P1	CPU	U35	2
97	278	10.000	17	==>	SA0	P1	CPU	U35	2
98	197	10.000	8	==>	SA0	P1	CPU	U35	2
100	279	10.000	19	==>	SA0	P1	CPU	U35	2
101	261	31.620	4	==>	SA1	P1	CPU	U35	2
102	224	31.620	107	==>	SA1	P1	CPU	U35	2
103	215	31.620	1	==>	SA1	P1	CPU	U35	2
104	278	31.620	17	==>	SA1	P1	CPU	U35	2
105	251	31.620	27	==>	SA1	P1	CPU	U35	2
106	284	31.620	23	==>	SA0	P1	CPU	U35	2
107	281	31.620	128	-81+R	SA0	P1	CPU	U35	2
108	283	31.620	22	==>	SA0	P1	CPU	U35	2
109	243	31.620	19	==>	SA0	P1	CPU	U35	2
110	248	31.620	24	-1+R	SA0	P1	CPU	U35	2
111	229	100.000	5	==>	SA1	P1	CPU	U35	2
112	254	100.000	2	-1+R	SA1	P1	CPU	U35	2
113	...	100.000	20	99	SA1	P1	CPU	U35	2
114	272	100.000	11	-47+R	SA1	P1	CPU	U35	2
115	246	100.000	22	-1+R	SA1	P1	CPU	U35	2
116	291	100.000	4	-108+R	SA0	P1	CPU	U35	2
117	260	100.000	2	==>	SA0	P1	CPU	U35	2
118	290	100.000	3	-1+R	SA0	P1	CPU	U35	2
119	246	100.000	22	==>	SA0	P1	CPU	U35	2
120	214	100.000	25	==>	SA0	P1	CPU	U35	2
121	266	316.220	5	==>	SA1	P1	CPU	U35	2
122	249	316.220	25	-1+R	SA1	P1	CPU	U35	2
123	287	316.220	26	==>	SA1	P1	CPU	U35	2
124	253	316.220	2	==>	SA1	P1	CPU	U35	2
125	209	316.220	20	==>	SA1	P1	CPU	U35	2
126	247	316.220	23	==>	SA0	P1	CPU	U35	2
127	274	316.220	14	==>	SA0	P1	CPU	U35	2
128	188	316.220	3	==>	SA0	P1	CPU	U35	2
129	288	316.220	2	==>	SA0	P1	CPU	U35	2
130	254	316.220	3	==>	SA0	P1	CPU	U35	2
131	242	1000.000	18	==>	SA1	P1	CPU	U35	2
132	248	1000.000	24	==>	SA1	P1	CPU	U35	2
133	245	1000.000	21	-1+R	SA1	P1	CPU	U35	2
134	209	1000.000	20	==>	SA1	P1	CPU	U35	2
135	282	1000.000	21	-72+R	SA1	P1	CPU	U35	2
136	214	1000.000	25	==>	SA0	P1	CPU	U35	2
137	239	1000.000	15	==>	SA0	P1	CPU	U35	2
138	251	1000.000	27	==>	SA0	P1	CPU	U35	2
139	186	1000.000	3	-1+R	SA0	P1	CPU	U35	2

140	251	1000.000	27	−1+R	SA0	P1	CPU	U35	2
161	...	0.001	...	...	SA1	P1	CPU	U38	27
162	...	0.001	...	...	SA1	P1	CPU	U38	27
163	...	0.001	2	2	SA1	P1	CPU	U38	27
164	246	0.001	22	==>	SA1	P1	CPU	U38	27
165	...	0.001	...	...	SA1	P1	CPU	U38	27
166	...	0.001	...	...	SA0	P1	CPU	U38	27
167	...	0.001	...	...	SA0	P1	CPU	U38	27
168	...	0.001	...	...	SA0	P1	CPU	U38	27
169	...	0.001	...	...	SA0	P1	CPU	U38	27
170	...	0.001	...	...	SA0	P1	CPU	U38	27
171	...	0.003	...	...	SA1	P1	CPU	U38	27
172	...	0.003	...	...	SA1	P1	CPU	U38	27
173	...	0.003	...	...	SA1	P1	CPU	U38	27
174	...	0.003	...	...	SA1	P1	CPU	U38	27
175	188	0.003	2	−33+R	SA1	P1	CPU	U38	27
176	...	0.003	...	...	SA0	P1	CPU	U38	27
177	...	0.003	...	...	SA0	P1	CPU	U38	27
178	...	0.003	...	...	SA0	P1	CPU	U38	27
179	...	0.003	...	...	SA0	P1	CPU	U38	27
180	...	0.003	...	...	SA0	P1	CPU	U38	27
181	234	0.010	10	==>	SA1	P1	CPU	U38	27
182	274	0.010	13	==>	SA1	P1	CPU	U38	27
183	...	0.010	...	...	SA1	P1	CPU	U38	27
184	...	0.010	...	...	SA1	P1	CPU	U38	27
185	...	0.010	...	...	SA1	P1	CPU	U38	27
186	...	0.010	...	...	SA0	P1	CPU	U38	27
187	...	0.010	...	...	SA0	P1	CPU	U38	27
188	...	0.010	...	...	SA0	P1	CPU	U38	27
189	...	0.010	...	...	SA0	P1	CPU	U38	27
190	...	0.010	...	...	SA0	P1	CPU	U38	27
191	201	0.032	12	−1+R	SA1	P1	CPU	U38	27
192	234	0.032	9	−108+R	SA1	P1	CPU	U38	27
193	197	0.032	8	−1+R	SA1	P1	CPU	U38	27
194	279	0.032	18	−108+R	SA1	P1	CPU	U38	27
195	...	0.032	...	...	SA1	P1	CPU	U38	27
196	...	0.032	...	...	SA0	P1	CPU	U38	27
197	...	0.032	...	...	SA0	P1	CPU	U38	27
198	...	0.032	3	3	SA0	P1	CPU	U38	27
199	...	0.032	...	...	SA0	P1	CPU	U38	27
200	...	0.032	...	...	SA0	P1	CPU	U38	27
201	192	0.100	3	==>	SA1	P1	CPU	U38	27
202	272	0.100	11	==>	SA1	P1	CPU	U38	27
203	289	0.100	106	−1+R	SA1	P1	CPU	U38	27
204	243	0.100	19	−1+R	SA1	P1	CPU	U38	27
205	214	0.100	132	==>	SA1	P1	CPU	U38	27
206	236	0.100	12	−1+R	SA0	P1	CPU	U38	27
207	...	0.100	...	...	SA0	P1	CPU	U38	27
208	...	0.100	...	...	SA0	P1	CPU	U38	27
209	...	0.100	...	...	SA0	P1	CPU	U38	27
210	...	0.100	...	...	SA0	P1	CPU	U38	27
211	...	0.316	...	...	SA1	P1	CPU	U38	27
212	257	0.316	3	==>	SA1	P1	CPU	U38	27
213	...	0.316	...	...	SA1	P1	CPU	U38	27
214	273	0.316	13	==>	SA1	P1	CPU	U38	27

215	186	0.316	3	-1+R	SA1	P1	CPU	U38	27
216	...	0.316	...	...	SA0	P1	CPU	U38	27
217	...	0.316	...	...	SA0	P1	CPU	U38	27
218	...	0.316	...	...	SA0	P1	CPU	U38	27
219	...	0.316	14	14	SA0	P1	CPU	U38	27
220	...	0.316	...	...	SA0	P1	CPU	U38	27
221	226	1.000	2	==>	SA1	P1	CPU	U38	27
222	217	1.000	2	-1+R	SA1	P1	CPU	U38	27
223	290	1.000	0	==>	SA1	P1	CPU	U38	27
224	241	1.000	16	-1+R	SA1	P1	CPU	U38	27
225	261	1.000	4	==>	SA1	P1	CPU	U38	27
226	...	1.000	...	...	SA0	P1	CPU	U38	27
227	282	1.000	21	==>	SA0	P1	CPU	U38	27
228	...	1.000	...	...	SA0	P1	CPU	U38	27
229	253	1.000	2	==>	SA0	P1	CPU	U38	27
230	288	1.000	2	==>	SA0	P1	CPU	U38	27
231	207	3.160	18	==>	SA1	P1	CPU	U38	27
232	255	3.160	4	==>	SA1	P1	CPU	U38	27
233	288	3.160	1	==>	SA1	P1	CPU	U38	27
234	254	3.160	2	-1+R	SA1	P1	CPU	U38	27
235	261	3.160	4	==>	SA1	P1	CPU	U38	27
236	267	3.160	6	-1+R	SA0	P1	CPU	U38	27
237	289	3.160	2	-1+R	SA0	P1	CPU	U38	27
238	257	3.160	3	==>	SA0	P1	CPU	U38	27
239	282	3.160	128	==>	SA0	P1	CPU	U38	27
240	272	3.160	11	==>	SA0	P1	CPU	U38	27
241	204	10.000	15	==>	SA1	P1	CPU	U38	27
242	250	10.000	26	==>	SA1	P1	CPU	U38	27
244	260	10.000	2	-1+R	SA1	P1	CPU	U38	27
245	286	10.000	26	-81+R	SA1	P1	CPU	U38	27
246	284	10.000	23	-6+R	SA0	P1	CPU	U38	27
247	254	10.000	3	==>	SA0	P1	CPU	U38	27
248	237	10.000	12	-1+R	SA0	P1	CPU	U38	27
249	269	10.000	7	-34+R	SA0	P1	CPU	U38	27
250	204	10.000	122	-1+R	SA0	P1	CPU	U38	27
251	235	31.620	10	-1+R	SA1	P1	CPU	U38	27
252	...	31.620	25	282	SA1	P1	CPU	U38	27
253	284	31.620	23	==>	SA1	P1	CPU	U38	27
254	187	31.620	4	-1+R	SA1	P1	CPU	U38	27
256	227	31.620	3	==>	SA0	P1	CPU	U38	27
257	184	31.620	2	==>	SA0	P1	CPU	U38	27
258	249	31.620	25	-72+R	SA0	P1	CPU	U38	27
259	250	31.620	26	==>	SA0	P1	CPU	U38	27
260	288	31.620	1	==>	SA0	P1	CPU	U38	27
261	...	0.001	...	...	SA1	P1	MPM2	U34	3
262	253	0.001	106	==>	SA1	P1	MPM2	U34	3
263	...	0.001	...	...	SA1	P1	MPM2	U34	3
264	...	0.001	...	...	SA1	P1	MPM2	U34	3
265	...	0.001	...	...	SA1	P1	MPM2	U34	3
266	...	0.001	...	...	SA0	P1	MPM2	U34	3
267	...	0.001	...	...	SA0	P1	MPM2	U34	3
268	...	0.001	...	...	SA0	P1	MPM2	U34	3
269	...	0.001	...	...	SA0	P1	MPM2	U34	3
270	...	0.001	...	...	SA0	P1	MPM2	U34	3
271	...	0.003	...	...	SA1	P1	MPM2	U34	3

272	...	0.003	...	...	SA1	P1	MPM2	U34	3
273	...	0.003	...	...	SA1	P1	MPM2	U34	3
274	253	0.003	2	==>	SA1	P1	MPM2	U34	3
275	216	0.003	1	==>	SA1	P1	MPM2	U34	3
276	244	0.003	20	==>	SA0	P1	MPM2	U34	3
277	961	0.003	772	==>	SA0	P1	MPM2	U34	3
278	222	0.003	5	==>	SA0	P1	MPM2	U34	3
279	...	0.003	36	1393	SA0	P1	MPM2	U34	3
280	...	0.003	...	...	SA0	P1	MPM2	U34	3
281	289	0.010	34	-1+R	SA1	P1	MPM2	U34	3
282	...	0.010	...	...	SA1	P1	MPM2	U34	3
284	188	0.010	2	-1+R	SA1	P1	MPM2	U34	3
285	274	0.010	13	==>	SA1	P1	MPM2	U34	3
286	...	0.010	...	...	SA0	P1	MPM2	U34	3
287	215	0.010	26	-1+R	SA0	P1	MPM2	U34	3
288	212	0.010	23	==>	SA0	P1	MPM2	U34	3
289	...	0.010	...	...	SA0	P1	MPM2	U34	3
290	...	0.010	...	...	SA0	P1	MPM2	U34	3
291	...	0.032	...	...	SA1	P1	MPM2	U34	3
292	388	0.032	199	==>	SA1	P1	MPM2	U34	3
294	...	0.032	302	380	SA1	P1	MPM2	U34	3
295	222	0.032	1	==>	SA1	P1	MPM2	U34	3
296	392	0.032	203	==>	SA0	P1	MPM2	U34	3
297	246	0.032	22	==>	SA0	P1	MPM2	U34	3
298	258	0.032	3	-1+R	SA0	P1	MPM2	U34	3
299	216	0.032	1	==>	SA0	P1	MPM2	U34	3
300	...	0.032	...	...	SA0	P1	MPM2	U34	3
301	...	0.100	...	...	SA1	P1	MPM2	U34	3
302	218	0.100	1	==>	SA1	P1	MPM2	U34	3
303	...	0.100	...	...	SA1	P1	MPM2	U34	3
304	254	0.100	3	==>	SA1	P1	MPM2	U34	3
305	...	0.100	...	...	SA1	P1	MPM2	U34	3
306	252	0.100	0	==>	SA0	P1	MPM2	U34	3
307	213	0.100	24	-107+R	SA0	P1	MPM2	U34	3
308	356	0.100	167	==>	SA0	P1	MPM2	U34	3
309	220	0.100	3	==>	SA0	P1	MPM2	U34	3
310	280	0.100	19	-1+R	SA0	P1	MPM2	U34	3
311	255	0.316	3	-1+R	SA1	P1	MPM2	U34	3
312	...	0.316	...	...	SA1	P1	MPM2	U34	3
313	...	0.316	...	...	SA1	P1	MPM2	U34	3
314	219	0.316	4	-1+R	SA1	P1	MPM2	U34	3
315	...	0.316	...	...	SA1	P1	MPM2	U34	3
316	241	0.316	17	-1+R	SA0	P1	MPM2	U34	3
318	257	0.316	3	==>	SA0	P1	MPM2	U34	3
319	235	0.316	11	==>	SA0	P1	MPM2	U34	3
321	270	1.000	9	==>	SA1	P1	MPM2	U34	3
322	251	1.000	27	==>	SA1	P1	MPM2	U34	3
323	250	1.000	4	==>	SA1	P1	MPM2	U34	3
324	...	0.010	...	...	SA1	P1	MPM2	U34	3
325	254	0.316	3	==>	SA0	P1	MPM2	U34	3
326	268	1.000	6	-48+R	SA1	P1	MPM2	U34	3
327	1091	1.000	9	==>	SA1	P1	MPM2	U34	3
329	252	1.000	1	==>	SA0	P1	MPM2	U34	3
330	236	1.000	12	-72+R	SA0	P1	MPM2	U34	3
331	254	1.000	3	==>	SA0	P1	MPM2	U34	3

333	243	1.000	19	==>	SA0	P1	MPM2	U34	3
334	227	3.162	3	-1+R	SA1	P1	MPM2	U34	3
335	237	3.162	13	==>	SA1	P1	MPM2	U34	3
336	199	3.162	10	==>	SA1	P1	MPM2	U34	3
337	251	3.162	27	-1+R	SA1	P1	MPM2	U34	3
355	...	0.032	...	...	SA1	P1	MPM2	U34	3

APPENDIX C

SURE Model

```
LAMBDA = 1E-4;
K = 10.0;
GAMMA = K*LAMBDA;
MU_W = 1E-9 TO* 1E-1 BY 10;
BETA = 2*MU_W;

W0 = 0.0;           W1 = 1E-6;          W2 = 3.16E-6;       W3 = 1E-5;
W4 = 31.62E-6;      W5 = 100.0E-6;      W6 = 316.22E-6;     W7 = 1.0E-3;
W8 = 3.162E-3;      W9 = 10E-3;         W10 = 31.62E-3;     W11 = 100E-3;
W12 = 316.22E-3;    W13 = 1000.0E-3;

ERW0 = 239.0;       ERW1 = 239.0;       ERW2 = 350.889;     ERW3 = 239.445;
ERW4 = 255.875;     ERW5 = 247.150;     ERW6 = 238.333;     ERW7 = 284.320;
ERW8 = 256.916;     ERW9 = 249.111;     ERW10 = 248.444;    ERW11 = 255.778;
ERW12 = 251.500;    ERW13 = 236.700;

ER2W0 = 57282.6;    ER2W1 = 57282.602;  ER2W2 = 181682.0;   ER2W3 = 58599.2;
ER2W4 = 68679.1;    ER2W5 = 62566.7;    ER2W6 = 57448.1;    ER2W7 = 108444.4;
ER2W8 = 69975.8;    ER2W9 = 62921.8;    ER2W10 = 62708.1;   ER2W11 = 66016.7;
ER2W12 = 64170.5;   ER2W13 = 56681.3;

EZW0 = 0.0;         EZW1 = 0.1;         EZW2 = 66.333;      EZW3 = 0.0;
EZW4 = 47.286;      EZW5 = 0.0;         EZW6 = 2.462;       EZW7 = 0.0;
EZW8 = 0.0;         EZW9 = 0.0;         EZW10 = 282.000;    EZW11 = 99.000;
EZW12 = 0.0;        EZW13 = 0.0;

EZ2W0 = .160;       EZ2W1 = .160;       EZ2W2 = 92402.3;    EZ2W3 = 0.0;
EZ2W4 = 15875.0;    EZ2W5 = 0.0;        EZ2W6 = 40.00;      EZ2W7 = 0.0;
EZ2W8 = 0.0;        EZ2W9 = 0.0;        EZ2W10 = 79524.0;   EZ2W11 = 9801.0;
EZ2W12 = 0.0;                           EZ2W13 = 0.0;


PRW0 = 0.0;         PRW1 = .17;         PRW2 = .30;         PRW3= .37;
PRW4= .53;          PRW5 = .69;         PRW6= .54;          PRW7 = .86;
PRW8 = 1.00;        PRW9 = 1.00;        PRW10 = .95;        PRW11 = .90;
PRW12 = 1.00;       PRW13 = 1.00;

FW0 = 1;     IF W0 < BETA  THEN FW0 = W0/BETA;
FW1 = 1;     IF W1 < BETA  THEN FW1 = W1/BETA;
FW2 = 1;     IF W2 < BETA  THEN FW2 = W2/BETA;
FW3 = 1;     IF W3 < BETA  THEN FW3 = W3/BETA;
FW4 = 1;     IF W4 < BETA  THEN FW4 = W4/BETA;
FW5 = 1;     IF W5 < BETA  THEN FW5 = W5/BETA;
FW6 = 1;     IF W6 < BETA  THEN FW6 = W6/BETA;
FW7 = 1;     IF W7 < BETA  THEN FW7 = W7/BETA;
FW8 = 1;     IF W8 < BETA  THEN FW8 = W8/BETA;
FW9 = 1;     IF W9 < BETA  THEN FW9 = W9/BETA;
FW10 = 1;    IF W10 < BETA  THEN FW10 = W10/BETA;
```

```
FW11 = 1;   IF W11 < BETA  THEN FW11 = W11/BETA;
FW12 = 1;   IF W12 < BETA  THEN FW12 = W12/BETA;
FW13 = 1;   IF W13 < BETA  THEN FW13 = W13/BETA;

MU_R =   ( FW1  - FW0 ) * ERW1 +   ( FW2  - FW1 ) * ERW2 +
         ( FW3  - FW2 ) * ERW3 +   ( FW4  - FW3 ) * ERW4 +
         ( FW5  - FW4 ) * ERW5 +   ( FW6  - FW5 ) * ERW6 +
         ( FW7  - FW6 ) * ERW7 +   ( FW8  - FW7 ) * ERW8 +
         ( FW9  - FW8 ) * ERW9 +   ( FW10 - FW9 ) * ERW10 +
         ( FW11 - FW10 ) * ERW11 + ( FW12 - FW11 ) * ERW12 +
         ( FW13 - FW12 ) * ERW13 ;


SIGMA_R = SQRT(
                ( FW1  - FW0 ) * ER2W1 +   ( FW2  - FW1 ) * ER2W2 +
                ( FW3  - FW2 ) * ER2W3 +   ( FW4  - FW3 ) * ER2W4 +
                ( FW5  - FW4 ) * ER2W5  +  ( FW6  - FW5 ) * ER2W6  +
                ( FW7  - FW6 ) * ER2W7  +  ( FW8  - FW7 ) * ER2W8  +
                ( FW9  - FW8 ) * ER2W9  +  ( FW10 - FW9 ) * ER2W10  +
                ( FW11 - FW10 ) * ER2W11 + ( FW12 - FW11 ) * ER2W12 +
                ( FW13 - FW12 ) * ER2W13  - MU_R*MU_R);

MU_Z =
         ( FW1  - FW0 ) * EZW1 +   ( FW2  - FW1 ) * EZW2 +
         ( FW3  - FW2 ) * EZW3 +   ( FW4  - FW3 ) * EZW4 +
         ( FW5  - FW4 ) * EZW5 +   ( FW6  - FW5 ) * EZW6 +
         ( FW7  - FW6 ) * EZW7 +   ( FW8  - FW7 ) * EZW8 +
         ( FW9  - FW8 ) * EZW9 +   ( FW10 - FW9 ) * EZW10  +
         ( FW11 - FW10 ) * EZW11 + ( FW12 - FW11 ) * EZW12 +
         ( FW13 - FW12 ) * EZW13 ;


SIGMA_Z = SQRT(
                ( FW1  - FW0 ) * EZ2W1 + ( FW2  - FW1 ) * EZ2W2 +
                ( FW3  - FW2 ) * EZ2W3 + ( FW4  - FW3 ) * EZ2W4 +
                ( FW5  - FW4 ) * EZ2W5 + ( FW6  - FW5 ) * EZ2W6 +
                ( FW7  - FW6 ) * EZ2W7 + ( FW8  - FW7 ) * EZ2W8 +
                ( FW9  - FW8 ) * EZ2W9 + ( FW10 - FW9 ) * EZ2W10 +
                ( FW11 - FW10 ) * EZ2W11 + ( FW12 - FW11 ) * EZ2W12 +
                ( FW13 - FW12 ) * EZ2W13  - MU_Z*MU_Z);


P_R =
         ( FW1  - FW0 ) * PRW1 +   ( FW2  - FW1 ) * PRW2 +
         ( FW3  - FW2 ) * PRW3 +   ( FW4  - FW3 ) * PRW4 +
         ( FW5  - FW4 ) * PRW5 +   ( FW6  - FW5 ) * PRW6 +
         ( FW7  - FW6 ) * PRW7 +   ( FW8  - FW7 ) * PRW8 +
         ( FW9  - FW8 ) * PRW9 +   ( FW10 - FW9 ) * PRW10 +
         ( FW11 - FW10 ) * PRW11 + ( FW12 - FW11 ) * PRW12 +
         ( FW13 - FW12 ) * PRW13 ;
```

```
MU_RP = MU_R;
SIGMA_RP = SIGMA_R;

(* convert to hours *)

MS_PER_HOUR = 1E3*60*60;
MU_R = MU_R/MS_PER_HOUR ;
SIGMA_R = SIGMA_R /MS_PER_HOUR;
MU_Z = MU_Z/MS_PER_HOUR;
SIGMA_Z = SIGMA_Z/MS_PER_HOUR;
MU_RP= MU_RP/MS_PER_HOUR;
SIGMA_RP = SIGMA_RP/MS_PER_HOUR;

SHOW MU_R,SIGMA_R,MU_Z,SIGMA_Z,MU_RP,SIGMA_RP;

1,2 = 4*GAMMA;
2,3 = 3*GAMMA + 3*LAMBDA;
1,4 = 4*LAMBDA;
4,5 = 3*GAMMA;
2,5 = 3*LAMBDA;
4,6 = 3*LAMBDA + 3*GAMMA;
2,7 = <MU_R, SIGMA_R, P_R>;
2,1 = <MU_Z, SIGMA_Z, 1-P_R>;
4,7 = <MU_RP, SIGMA_RP>;
7,8 = 3*GAMMA;
8,9 = 2*GAMMA + 2*LAMBDA;
7,10  = 3*LAMBDA;
10,12 = 2*LAMBDA + 2*GAMMA;
10,11 = 2*GAMMA;
8,12  = 2*LAMBDA;
8,13 = <MU_R, SIGMA_R, P_R>;
10,13 = <MU_RP, SIGMA_RP>;
8,7 =  <MU_Z, SIGMA_Z, 1-P_R>;
13,14 = GAMMA + LAMBDA;
```

REFERENCES

1. McConnel, Stephen R.; Siewiorek, Daniel P.; Tsao, Michael M.: The Measurement and Analysis of Transient Errors in Digital Computer Systems, The Ninth Annual International Symposium on Fault-Tolerant Computing, June 20-22, 1979.

2. Goldberg, Jack; Kautz, William H.; Melliar-Smith, P. Michael; Green, Milton W.; Levitt, Karl N.; Schwartz, Richard L.; and Weinstock, Charles B.: Development and Analysis of the Software Implemented Fault Tolerance (SIFT) Computer. NASA CR-172146, 1984.

3. Miller, Douglas R: A Note on the Independence of Multivariate Lifetimes In Competing Risk Models. Annuals of Statistics, 1977, Vol. 5, No. 3, pp. 516-579.

4. Butler, Ricky W.: The Semi-Markov Unreliability Range Evaluator (SURE) Program. NASA TM-86261, 1984.

5. Shin, Kang G.; Woodbury, Michael W.; and Lee, Yang-Hang: Modeling and Measurement of Fault-Tolerant Multiprocessors. NASA CR-3920, August 1985.

6. Bavuso, S. J.; and Petersen, P. L.: CARE III Model Overview and User's Guide (First Revision). NASA TM-86404, 1985.

7. Lala, Jaynarayan H.; and Smith, T. Basil, III: Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer, Volume III - FTMP Test and Evaluation. NASA CR-166073, 1983.

Standard Bibliographic Page

1. Report No. NASA TM-89058	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Preliminary Transient-Fault Experiment on the SIFT Computer System		5. Report Date February 1987
		6. Performing Organization Code 505-66-21-01
7. Author(s) Ricky W. Butler and Carl R. Elks		8. Performing Organization Report No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225		10. Work Unit No.
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001		13. Type of Report and Period Covered Technical Memorandum
		14. Sponsoring Agency Code

15. Supplementary Notes

Ricky W. Butler, Langley Research Center, Hampton, Virginia.
Carl R. Elks, PRC Kentron, Inc., Hampton, Virginia (presently with U.S. Army Aerostructures Directorate, USAARTA (AVSCOM), Langley Research Center, Hampton, Virginia).

16. Abstract

This paper presents the results of a preliminary experiment to study the effectiveness of a fault-tolerant system's ability to handle transient faults. The primary goal of the experiment was to develop the techniques to measure the parameters needed for a reliability analysis of the SIFT computer system which includes the effects of transient faults. A key aspect of such an analysis is the determination of the effectiveness of the operating system's ability to discriminate between transient and permanent faults. A detailed description of the preliminary transient fault experiment along with the results from 297 transient fault injections are given. Although not enough data was obtained to draw statistically significant conclusions, the foundation has been laid for a large-scale transient fault experiment.

17. Key Words (Suggested by Authors(s)) Transient Faults Fault Tolerance Error Fault Latency Fault Injection	18. Distribution Statement Unclassified - Unlimited Star Category 62

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 42	22. Price A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA Langley Form 63 (June 1985)