Failure Map Functions and Accelerated Mean Time to Failure Tests: New Approaches for Improving the Reliability Estimation in Systems Exposed to Single Event Upsets

Pablo A. Ferreyra, Carlos A. Marqués, Ricardo T. Ferreyra, and Javier P. Gaspar

***Abstract*—The application cross section "σ_{AP}" is a parameter used to characterize the systems single event upset (SEU) vulnerability, but does not give, in a direct way, the system's reliability. Reliability prediction of a system exposed to SEU can be improved with the knowledge of the system's time to failure (TTF) probability distribution function. This work presents two new methods suitable to provide this information. The first one is based on the construction of a function named the failure map function (FMF). FMF contains the information needed to calculate all the TTF statistical properties by means of numerical procedures. For the cases where the FMF function is difficult to obtain, a second method is presented which consist of injecting SEUs at a rate several orders of magnitude higher than the real rate. A histogram of the TTF random variable for the accelerated process is obtained. The system's reliability can then be derived by means of statistical and numerical procedures.**

***Index Terms*—Fault tolerance, radiation environment, single event upsets (SEUs) computer reliability, single event upset (SEU) injection.**

I. Introduction

RELIABILITY prediction in digital systems exposed to single event upsets (SEUs) is a central aspect in the development of fault tolerant computers devoted to operate in radiation environments (e.g., space, nuclear power plants) and, with the miniaturization, becomes a concern for applications in the earth's atmosphere.

A device static cross-section, (σ_{D}), can be obtained by means of static radiation ground testing, [1]–[7]. Basically, the determination of σ_{D} for a memory register consists of counting the number of bit flips that take place while the register is being exposed to a particle flux. If the register is an internal microprocessor element, the determination of σ_{D} is more complicated. However, there exist methods that have been used to obtain σ_{D} for internal processor's memory elements. Also, there exist methods to define and to obtain the static cross section σ_{D} for the whole processor. Since an application running in the processor uses the processor registers only during a fraction of the time, σ_{D} is an over-estimation of the system's vulnerability to SEUs. A more accurate parameter is the application's cross section (σ_{AP}). The determination of σ_{AP} requires the use of the so-called *dynamic test*, which consist of exposing the device under study to a particle flux while a program as close as possible to the final application is executed by the processor. A suitable test platform must be used to implement such tests and count on-line the number of errors observed during the irradiation experiment, [8]. The main problem of this kind of test is that the results are strongly dependent on the particular application running in the processor. Hence, any minor change made in the program implies the need for a new dynamic test, resulting in a very high cost process. To tackle this problem, a three-step approach was recently presented, [7]. The first step consists in measuring the static cross-section, (σ_{D}), by means of static radiation ground testing. Note that σ_{D} is a characteristic of the processor under study and does not depend on the application. The second step uses fault injection experiments to obtain the average rate of program malfunctions per upset, called (τ) in the following. The determination of τ consists of injecting bit flips with a hardware-software scheme while the processor executes the application. A dedicated test platform must be used to generate bit flips and count on-line the number of errors observed. However, note that τ is obtained without irradiation, resulting in a low cost process. This allows the study of the effects caused by minor program changes. In the third step, the application cross-section, σ_{AP}, is estimated as the product $\sigma_{\mathrm{AP}} = \sigma_{\mathrm{D}} * \tau$. Hence, it is clear that, the approach presented in [7] avoids the need to perform dynamic radiation ground testing. However, the only reliability indicator obtained with the previous method is σ_{AP} and this parameter does not give, in a direct way, the system's reliability.

By definition system's reliability at time t is the probability that the system works according to the specifications until time t. The system's time to failure, (TTF), is a random variable. To calculate the system's reliability, the TTF cumulative distribution function must be obtained. In this work we present two new methods suitable to provide the systems TTF cumulative distribution function for processors included in a targeted architecture. As in [7] the strategies investigated in this paper comprise three steps, the first one being the determination of the static

Manuscript received September 30, 2003; revised April 10 and July 12, 2004.

P. A. Ferreyra is with the Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, Medina Allende y Haya de la Torre, Cuidad Universitaria, 5000 Córdoba, Argentina and also with the Facultad de Ingeniería, Instituto Universitario Aeronáutico, Córdoba, Argentina (e-mail: ferreyra@mail.famaf.unc.edu.ar).

C. A. Marqués, and J. P. Gaspar are with the Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, Medina Allende y Haya de la Torre, Cuidad Universitaria, 5000 Córdoba, Argentina.

R. T. Ferreyra is with the Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba, Argentina.

Digital Object Identifier 10.1109/TNS.2005.845883

0018-9499/$20.00 © 2005 IEEE

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.

cross-section figures by performing suitable radiation ground testing and the second step being based on fault injection experiments. The third step consists of obtaining the TTF distribution function by means of statistic and numeric procedures.

Our first method aims at obtaining a curve, which we will call "Application's Failure Map Function," by implementing an appropriate fault-injection failure-detection algorithm. This fault injection scheme can be implemented with different kind of simulation systems, that can range from software based simulation tools [9], [10], to real time fault injection schemes [11]. Once the Failure Map Function is obtained, the system's time to failure cumulative distribution function can be derived.

The second method produces a histogram of the TTF random variable obtained by means of a real time fault injection system. It is also possible to obtain the TTF histogram with a software based simulation tool, but the required simulation time may be on the order of months with presently available computational power. The rate at which SEU take place is accelerated, several orders of magnitude in order to reduce the required simulation time. The system's reliability can then be derived by means of statistical and numerical procedures.

The foundation of both methods consists of the association of suitable Poisson models [12] to the time instants where the radiation particles interact with the processor. The construction of such models is discussed in Sections II and III. Section IV presents an algorithm to obtain TTF probability distribution function given the FMF, and Section V presents a practical procedure for the implementation of accelerated mean time to failure tests. Section VI presents the application of the proposed methods to a FIR (Finite Impulse Response) filter routine for a digital signal processor.

II. Associated Poisson Models

There are three types of event that take place in a digital computer when it executes a program while it is being irradiated. The first kind of event is denoted as "type I" and defined as a radiation particle passage through the processor's sensitive area "$\mathrm{S_P}$." A "*type II*" *event* is a SEU taking place at any of the "N" processor's information storage bits. The third kind of event, denoted as "*type III*," is the start of an error propagation in the program execution due to a SEU.

It is well known from the physics field that when an area "$\mathrm{S_P}$" is being irradiated with an "f" intensity flux, the time instants where a radiation particle passage takes place can be modeled as a homogeneous Poisson Process. This kind of stochastic process is fully described by the "*particle arrival rate*" denoted by "λ" in the following. Parameter λ is related to the flux intensity f and the processor's sensitive area "$\mathrm{S_P}$" by means of

$$\lambda = f \cdot S_P. \tag{1}$$

The Poisson process theory allows the prediction of all the statistical properties of random variables associated to type I events based on the knowledge of the λ parameter. In particular, if "$\mathrm{T_P}$" denotes the random variable associated to the first type I event occurrence, measured from the processor's reset, then the cumulative probability distribution function of "$\mathrm{T_P}$" is given by

$$F_{T_P}(t) = 1 - e^{-\lambda \cdot t} \quad t \geq 0. \tag{2}$$

The mean value of $\mathrm{T_P}$ is given by

$$\overline{T_P} = \frac{1}{\lambda}. \tag{3}$$

Time instants associated with the occurrence of a type II event can also be modeled with a homogeneous Poisson process, which is fully described by the "*SEU rate*" parameter denoted in the following by "λ_S". Parameter λ_S is related to the λ parameter by means of (4), where "σ_P" denotes the processor's static cross section for the flux under consideration. Note that σ_P is obtained with static radiation tests and is independent of the application

$$\lambda_S = \lambda \cdot \frac{\sigma_P}{S_P}. \tag{4}$$

If "$\mathrm{T_S}$" denotes the random variable associated to the first type II event occurrence measured from the processor's reset, then the cumulative probability distribution function of "$\mathrm{T_S}$" and its mean value are given by (5) and (6), respectively

$$F_{T_S}(t) = 1 - e^{-\lambda_s \cdot t} \quad t \geq 0 \tag{5}$$

$$\overline{T_S} = \frac{1}{\lambda_S}. \tag{6}$$

It can be demonstrated that time instants associated with the occurrence of a type III event can be modeled with a nonhomogeneous Poisson process, which is fully described by the "*failure rate function*" denoted in the following by "$\lambda_F(t)$." To obtain $\lambda_F(t)$, the *conditional probability* that a type III event takes place at time t given that a type II event occurs at the same instant, should be determined. In the next section it will be shown that this probability is given by the "*failure map function*" denoted by $\mathrm{FMF}(t)$.

Once $\mathrm{FMF}(t)$ is known, $\lambda_F(t)$ is given by

$$\lambda_F(t) = \mathrm{FMF}(t) \cdot \lambda_S. \tag{7}$$

If "$\mathrm{T_F}$" denotes the random variable associated to the first type III event occurrence measured from the processor's reset, then the cumulative probability distribution function of "$\mathrm{T_F}$" is given by

$$F_{T_F}(t) = 1 - e^{\cdot \int_{x=0}^{x=t} \lambda_F(x)\,dx} = 1 - e^{\cdot \int_{x=0}^{x=t} \lambda_s \cdot \mathrm{FMF}(x)\,dx}. \tag{8}$$

A general expression for the mean value of $\mathrm{T_F}$ is given by

$$\overline{T_F} = \int_{x=0}^{x=\infty} e^{-\int_{u=0}^{u=x} \lambda_F(u)du}\,dx. \tag{9}$$

From the last two equations it is clear that all the statistical properties for the $\mathrm{T_F}$ random variable are completely determined by the function $\mathrm{FMF}(t)$.

III. Failure Map Functions (FMFs)

A synchronous digital computer is a set of combinational and sequential circuits under control of a common square wave signal named the processor's clock. State transitions take place only at the clock edges. Hence, a discrete type

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.

variable, $k, k = 1,2,\ldots,n$ can be associated to the n first clock signal edges. The variable k represents a time instant during a particular program execution. For example, $k = 0$, represents the beginning of program execution. Note that if a processor comprises N bits, (flip-flops, registers, internal SRAM and/or cache memories, etc.), the state of all processor bits at a certain time instant k, can be represented with an array $\mathrm{B}(k) = \{\mathrm{B}_0(k), \mathrm{B}_1(k), \ldots, \mathrm{B}_\mathrm{N}(k)\}$, where, $\mathrm{B}_i(k)$ represents the state of the bit "B_i" at time k. If the execution of a program needs N_P clock pulses to complete in the processor, then the corresponding process can be represented by the succession: $\mathrm{P} = \{(k, \mathrm{B}(k)/k = 1, \ldots, \mathrm{N}_P)\}$. The process P can be viewed as a two dimensional array, where the kth column is given by $\mathrm{B}(k)$. Note that for each time instant k, there are a certain number of bits of the array $\mathrm{B}(k)$, which are relevant, i.e., that contain useful information, for program execution. This number of relevant bits depend on the time instant k and is denoted by $\mathrm{N}_{\mathrm{re}}(k)$. There is also another number of bits that are completely irrelevant for the program execution. This number of irrelevant bits depend of the time instant k too, and are denoted by $\mathrm{N}_{\mathrm{irr}}(k)$. In general, it is possible to design a procedure to identify if a bit B_i is relevant or irrelevant for program execution at a particular instant k, by means of fault injection experiments. Based on this procedure, a new N element array denoted $\mathrm{FM}(k)$, can be defined for each time instant k in the following way: $\mathrm{FM}(k) = \{\mathrm{FM}_0(k), \mathrm{FM}_1(k), \ldots, \mathrm{FM}_\mathrm{N}(k)\}$, where $\mathrm{FM}_i(k) = 1$, if $\mathrm{B}_i(k)$ is relevant and $\mathrm{FM}_i(k) = 0$, if $\mathrm{B}_i(k)$ is irrelevant to the program execution. Again, if the program needs N_P clock pulses to complete, a two dimensional array denoted by "FMP", can be associated to the program execution, where the kth column of FMP is the array $\mathrm{FM}(k)$, and can be represented by the succession: $\mathrm{FMP} = \{(k, \mathrm{FM}(k)/k = 1, \ldots, \mathrm{N}_\mathrm{P})\}$. Note that FMP is an N by N_P array. The element $\mathrm{FMP}(i,k)$, is "1" if the status of the ith processor bit, at time k, is relevant for program execution and "0" if not. A Failure Map for the program under study is a two dimensional representation for the array FMP, in a Cartesian graph, where a dot is plotted in the coordinate (i,k), if $\mathrm{FMP}(i,k) = 1$, and is left blank if $\mathrm{FMF}(i,k) = 0$. It is important to note that the dots in the failure map represent points in time and space, where a SEU initiates an error propagation process that will affect the program execution.

$\mathrm{FMF}_i(t)$ denotes the FMF at time t for bit $\mathrm{B}i$. The value of $\mathrm{FMF}_i(t)$ is 1, if bit $\mathrm{B}i$ is relevant for the program execution at time t, and 0 if not. $\mathrm{FMF}_i(t)$ values can only change at the processor clock signal transitions and represent the conditional probability of a type III event originated in bit B_i at time t given the occurrence of a type II event at the same time t and bit.

It can be shown that $\mathrm{FMF}(t)$ i.e., the conditional probability that a type III event takes place at time t, given the occurrence of a type II event in any of the N processor bits, is given by (10), where "σ_i" denotes bit $\mathrm{B_i}$ cross section for the given flux

$$\mathrm{FMF}(t) = \frac{\sum_{i=1}^{i=N} \mathrm{FMF}_i(t) \cdot \sigma_i}{\sigma_P}. \tag{10}$$

Note that, since $\mathrm{FMF_i(t)}$ values, $\mathrm{i} = 1, \ldots, \mathrm{N}$, can only change at the processor clock signal transitions, FMF(t) inherits the same behavior.

Under the *simplifying assumption* that all internal processor bits have the same static cross section, $(\sigma_\mathrm{i} = \sigma_\mathrm{P}/\mathrm{N})$, (10) reduces to (11). In (11), "t" is any value in the interval $[\mathrm{k}{*}\mathrm{T_C}, (\mathrm{k}+1){*}\mathrm{T_C})$, where $\mathrm{T_C}$ is the processor clock signal period. (Note that the interval is closed at the left end but open at the right end).

$$\mathrm{FMF}(t) = \frac{N_{\mathrm{re}}(k)}{N_{\mathrm{re}}(k) + N_{\mathrm{irr}}(k)} = \frac{N_{\mathrm{re}}(k)}{N} \tag{11}$$

IV. FMF-Based TTF Distributution

The random variable $\mathrm{T_F}$ (cf. Section II) represents the time instants where an error propagation starts due to the occurrence of a SEU in the processor during program execution. In general, the resulting malfunction will appear at the system's outputs some time later. However, in real time applications this amount of time is negligible if compared to the mean $\mathrm{T_F}$ or a typical value. Hence, for a large set of real time applications, the $\mathrm{T_F}$ random variable is almost the same as the system's time to failure, (TTF) random variable. In this case (8) gives a direct numerical method to obtain the system's TTF cumulative probability distribution function. In the same way, (9) gives another direct numerical method for the system's mean time to failure MTTF. From both equations it is clear that the inputs for these numerical procedures are λ_S and FMF(t). Parameter λ_S is obtained from the flux intensity, the processor sensitive area $\mathrm{S_P}$ and the static cross section σ_P that the processor presents under the given flux. There are several methods to obtain σ_P by means of static radiation ground testing [1]–[7]. These methods have been used and improved for many years. For example, the golden chip method. Basically, the processor under test and the reference processor execute a test program simultaneously. The processor under test is exposed to a radiation flux. The outputs of the two chips are compared and differences recorded. A test program is designed to test data in a particular internal processor register. A set of test programs is used to test all the internal registers. Since each test program is very simple, the duty factors of each internal element can be obtained by inspection and the element static cross section can be calculated. The manufacturer should provide both internal circuitry functional information and the size to calculate σ_P for the whole processor. In most cases, neither a detailed knowledge of the device internal circuitry nor the sizes of the internal elements are available. However, static test programs can be modified in order to test all internal processor registers and modules and to obtain an average value for σ_P. Note that the value σ_P obtained is completely independent of the final application under study. The processor sensitive area $\mathrm{S_P}$ is also a manufacturer data. The value of $\mathrm{S_P}$ is easier to obtain or to estimate. The flux intensity is provided by environment data and can be reproduced with proper equipment. Once all the previous data is obtained, the SEU rate λ_S can be calculated. Again, note that the value λ_S obtained is independent of the final application. It is also important to note that the values $\sigma_\mathrm{P}, \mathrm{S_P}$, and the flux intensity are known inputs.

To obtain $\mathrm{FMF}(t)$ with (10), the required inputs are $\mathrm{FMF}_i(t), \sigma_\mathrm{i}, i = 1, \ldots, \mathrm{N}$ and σ_P. To obtain the exact shape of the functions $\mathrm{FMF_i(t)}, \mathrm{i} = 1, \ldots, \mathrm{N}$, a description of the

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.

processor's internal circuitry is required. As previously stated, in most cases, this information is not available. However, it is always possible to construct a register transfer level (RTL), i.e., a model of the processor under study, based on the processor's data sheet and the instruction set format. Of course, if the internal circuitry is known a more accurate RTL model can be constructed. Still, the information contained in the processor data sheets and the instruction set format is enough to construct a suitable RTL model. Once the processor's RTL model is constructed, a software simulator tool (SST) can be developed with a high level language. The SST must accept as input the binary code and data generated by the available compilers and linkers for the processor. If the processor RTL model has N internal bits, the processor's state at a time instant "k" can be stored in a N element array $\mathrm{B(k)} = \{\mathrm{B_0(k)}, \mathrm{B_1(k)}, \ldots, \mathrm{B_N(k)}\}$, as stated in Section III. If the execution of the application under study needs $\mathrm{N_P}$ clock pulses to complete in the processor model, then the succession $\mathrm{P} = \{(\mathrm{k}, \mathrm{B(k)/k} = 1, \ldots, \mathrm{N_P})\}$ of all the processor's bit states can be stored in a file. As stated in Section III, this file can be viewed as a two dimensional matrix. This file is the *reference pattern* for the application under study. The SST allows running the RTL model and generating the reference pattern file. SST also allows to run the RTL model and invert the information of bit $\mathrm{B_i}$ at time instant k. The corresponding output file is named de (i,k) *pattern*. The functions $\mathrm{FMF_i(t)}, \mathrm{i} = 1, \ldots, \mathrm{N}$ can be obtained by comparing the reference pattern with the (i,k) patterns. The functions $\mathrm{FMF}_i(t), i = 1, \ldots, \mathrm{N}$ for the application under study, can also be obtained with software or hardware fault injection-failure detection schemes of the same type as those described in [9], [10], and [11] respectively. However, in these last two cases, it is only possible to generate a restricted set of (i,k) patterns, and hence only an approximation is obtained.

The functions $\mathrm{FMF_i(t)}, \mathrm{i} = 1, \ldots, \mathrm{N}$, depend only on the considered application and are independent of the parameters $\sigma_\mathrm{P}, \mathrm{S_P}, \sigma_\mathrm{i}, \mathrm{i} = 1, \ldots, \mathrm{N}$, and the flux intensity. Moreover, the effort in the development of the required fault-injection failure detection schemes is similar to those used in the application cross section approach described in [7].

The cross section values $(\sigma_\mathrm{i}, i = 1, \ldots, \mathrm{N})$ of all the processor's internal sensitive nodes should be provided by means of radiation tests. However, since it is easier to obtain the experimental value σ_P for all the processors as an average, rather than the individual values σ_i for all the internal sensitive nodes, it is easier to use (11) than (10) for the determination of the $\mathrm{FMF}(t)$ function. Moreover, the values $\sigma_\mathrm{i}\ \mathrm{i} = 1, \ldots, \mathrm{N}$ can be considered random variables with mean value $(\sigma_\mathrm{P}/\mathrm{N})$. This implies that if the number of relevant bits is large, (11) is a very good approximation for (10).

Hence, the procedure to obtain the cumulative distribution function of the TTF random variable, consists in the three following steps.

1) Obtain λ_S parameter using (4), with σ_P, S_P and the flux intensity as inputs.
2) Find the failure map function $\mathrm{FMF}(t)$, using (10) if values $\sigma_i\ i = 1, \ldots, N$ are known or using (11) if only σ_P is known.
3) Calculate the integral given by (8) for a set of t values, and use interpolation.

If experimental data for the λ_S parameter are available, the only required task is to find the application's failure map function. A possible limitation is the maximum t needed to evaluate $\mathrm{FMF}(t)$ in (8) and (9). However, in the case that a processor executes an endless loop repeating the same process, $\mathrm{FMF}(t)$ results in a periodic function, (except for the startup code), and hence $\mathrm{FMF}(t)$ is defined for any t. When that processor executes an endless loop but does not repeat continuously the same procedure, we only need to evaluate $\mathrm{FMF}(t)$, until a value of t, where the error in the integral of (8) (or (9)) results are negligible due to the exponential decay. Moreover, if the process under study is a series of different programs, $\mathrm{FMF}(t)$ can be obtained by means of a concatenation of the FMF functions of the individual programs. For most applications, a value for "T" can be found such that the $\mathrm{FMF(t)}$ function has a mean value, that is constant when computed in any time intervals of length T. This class of applications are called in this work: "*stationary over T*", and the interval T is called the "*stationary period*". Note that if $\mathrm{FMF}(t)$ is periodic with period T, it is also stationary over T, but the opposite is not true. Given a $\mathrm{FMF}(t)$ that is stationary over T, any interval of length T is in some sense representative of $\mathrm{FMF}(t)$. A periodic function $\mathrm{FMF}^*(t)$ can be constructed by means of the concatenation or repetition of a particular period of length T of $\mathrm{FMF}(t)$. $\mathrm{FMF}^*(t)$ is called in this work a "*periodic derivation*" of $\mathrm{FMF}(t)$. It can not be proven that the TTF cumulative distribution function is the same for $\mathrm{FMF}(t)$ and $\mathrm{FMF}^*(t)$, but under certain assumptions, it can be shown that the Mean Time To Failure (MTTF) has the same value.

V. Accelerated MTTF Tests

The concatenation procedure of the FMF functions described above may be very difficult to accomplish for certain applications. For example a computer operating system, which is a complex application. The kernel is continuously checking status flags and different routine sequences are called in response to different events. Clearly the operating system is not a periodic application in the sense discussed in the previous section. However, it can be shown that there exist a value for T where the application is stationary over T, (see the previous section). For this kind of applications, the accelerated MTTF test provides a simple solution, as discussed in this section. A mean time to failure test consists of injecting SEUs to the system under study at random time instants and target bits, and checking the system's functional status to determine the instant of a failure. The process is repeated in order to obtain a histogram of the system's time to failure data. As discussed before, SEU occurrence instants in a processor can be modeled with a homogeneous Poisson process with arrival rate $\lambda_\mathrm{S} = \mathrm{f} * \sigma_\mathrm{P}$. The idea in a MTTF test is to mimic a real situation. To do so, a software simulator tool can be used, but the required simulation times are extremely large and can result in the order of months with the computational power available today. In practice, the method can only be implemented with a real time or quasi real time fault injection system as the one described in [11]. However, the method may be implemented in the near future with a

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.

software simulator, if the computational power available today is increased by at least two orders of magnitude. Since the real values of σ_{P} are very small, the SEU arrival rates are also very small quantities and the required simulation time for real-time fault injection systems is huge. Since λ_{S} is proportional to λ, the simulation time may be reduced if the parameter λ is increased. If λ_0 denotes a normalized λ value any other λ value can be obtained multiplying λ_0 by an *acceleration factor* $\alpha(\lambda = \lambda_0 * \alpha)$. When the values of λ are very small, (small acceleration factor) and the application is *stationary*, it can be demonstrated that the mean value for the random variable $\mathrm{T_F}$ given by (9), is equal to the MTTF* value given by (12). The $\mathrm{T_F}$ random variable mean value is the system's mean time to failure denoted in the following by MTTF. In (12), the denominator is the mean value of the failure rate function $\lambda_{\mathrm{F}}(t)$ (cf. (7)). Hence (12) can be rewritten as (13). When the values of λ are very high, (high acceleration factor) the value MTTF* given by (12) is no longer equal to the MTTF value given by (9)

$$\mathrm{MTTF}^* = \frac{1}{\lambda_{t\to\infty}\frac{\int_{x=0}^{x=t}\lambda_F(x)dx}{t}} \tag{12}$$

$$\mathrm{MTTF}^* = \frac{1}{\overline{\lambda_F}}. \tag{13}$$

However, the MTTF value given by (9) is equal to the MTTF* value given by (12) for a large range of λ values. This fact can be shown plotting a curve of the quotient (MTTF/MTTF*) for a large range of acceleration factors α.

If the application is periodic with period T, the mean value of the failure rate function $\lambda_{\mathrm{F}}(t)$ is very easy to obtain. Hence, for periodic applications, the MTTF* value is also easy to obtain with (13). For periodic applications it is also possible to obtain the MTTF value given by (9), by means of a numerical procedure. If the application is not periodic but is still stationary over T, we can construct a periodic derivation and apply (13), to obtain the MTTF* value. Also, the MTTF value given by (9) can be obtained by means of a numerical procedure. Hence, for both type of application, (periodic with period T and stationary over T), it is possible to obtain a curve of the quotient (MTTF/MTTF*) as a function of λ or α values.

A typical (MTTF/MTTF*) quotient curve, (see Fig. 4 in the next section), has a flat zone where the quotient equals to 1. In the flat zone, MTTF and MTTF* values are equal. The important point here is that for all the λ values in the flat zone, the actual shape of the failure map function is irrelevant to obtain the MTTF parameter. In other words, in the quotient curve flat zone, the MTTF value depends only on the failure's map function mean value. A stationary application will have the same MTTF than any of its periodic derivations in the flat zone. Hence, any periodic derivation can be used to obtain the maximum λ or α value for which the failure map function shape is irrelevant to obtain the MTTF value. These maximum values will be denoted by λ_{MAX} and α_{MAX}.

It is clear that α_{MAX} is the highest acceleration factor that can be used to perform an accelerated mean time to failure test.

The previous discussion gives a numerical procedure for the calculation of the highest acceleration factor α_{MAX} that can be adopted. Using the periodic derivation we find the range of values for which the quotient (MTTF/MTTF*) is flat and equal to 1, and then simply use the maximum value of λ in that region to perform the MTTF accelerated test. This value for λ is generally several orders of magnitudes higher than the original value given by the real values of the flux and the device cross-section. Hence, a practical procedure to obtain the cumulative distribution function for the TTF random variable, for a stationary, non periodic application where the FMF function is difficult to obtain for every t, can be stated as follows:

1) Construct a periodic derivation for the application (FMF*).
2) Find the largest acceleration factor α_{MAX} for which the quotient (MTTF/MTTF*) curve is flat.
3) Use the corresponding λ_{MAX} and generate a TTF histogram with an accelerated MTTF test.

Once the histogram for the accelerated MTTF test is obtained for a given acceleration factor, the real histogram can be calculated with a homothetic transformation.

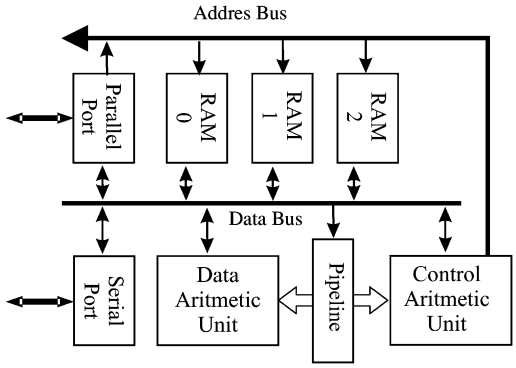

Fig. 1. DSP32C digital signal processor block diagram.

VI. Case Study

To illustrate the advantages and feasibility of the proposed methods, a finite impulse response (FIR) filter routine implemented in a digital signal processor (DSP) was studied. The processor used is a DSP32C from AT&T. Fig. 1 is a block diagram for the DSP32C. By means of direct memory access requests (DMA) it is possible to access any bit of the internal RAM modules. A host computer system can use the DSP32C parallel port interface to perform a DMA operation without disturbing the application running in the DSP32C. The FIR routine and data is stored in the internal RAM modules. The DMA access is performed in parallel with program execution. This means that with a proper interface, it is possible to simulate SEU almost in real time, (see [11]).

The DSP32C support software library package includes a simulator, (D3SIM). A software tool to obtain the $\mathrm{FMF_i(t)}$, functions for the FIR application, was developed (see [9], [10]). Fig. 2, is a partial failure map where the vertical axis represents an internal RAM memory area where some of the variables used by the program are located. Horizontal axis represents time measured in processor clock units. It is interesting to note that the figure provides in a simple way useful information about the vulnerable area of the programs (dark area).

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.

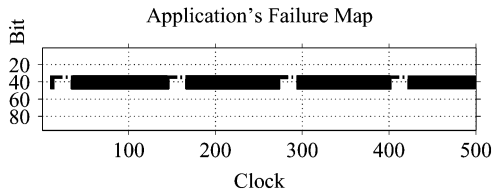

Fig. 2. A partial failure map for the FIR filter routine running in the DSP32C digital signal processor.

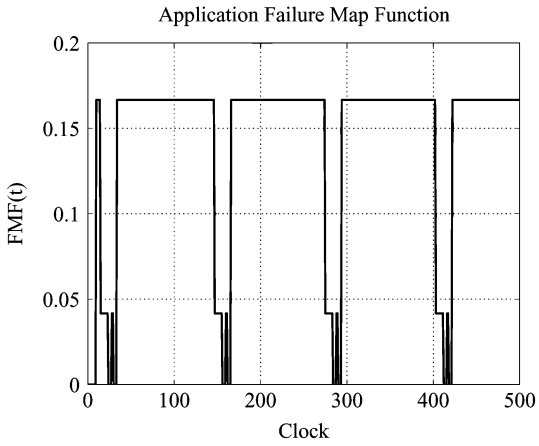

Fig. 3. The associated FMF function for the FIR filter in a digital signal processor, assuming a normalized value for λ_S.

This has a very important application in the design of fault tolerant systems, because, the problematic bit and instants are identified very precisely and rapidly. Fig. 3 is the associated failure map function for the failure map of Fig. 2. The failure map function was obtained by means of (11). Taking a close look to Fig. 3 it is easy to conclude that except for the start up part of the execution, the $\mathrm{FMF}(t)$ function is periodic and well defined for every t. Hence, the TTF distribution function can be obtained from a numerical computation of the integral in (8). Next, the application cross-section can be derived. The MTTF can also be calculated directly from the FMF function curve. Since the studied application is periodic, and it is easy to obtain $\mathrm{FMF}(t)$ for every t, there is no need to apply the MTTF accelerated test procedure. (Note that in this example a normalized value for λ_S was used).

However, if we assume that the application is not periodic but the time interval shown in Figs. 2 or 3 is the stationary period T for the application under study, then we can proceed as follows:

1) The periodic derivation for the application (FMF*) is obtained by means of a simple numerical procedure. The failure map function is a table stored in a file, and hence it is very easy to design a software procedure that acts as the FMF* function.
2) Find the largest acceleration factor α_{MAX}. Fig. 4 is the (MTTF/MTTF*) quotient curve, (see the previous section). The curve is plotted for a large range of acceleration factors. That is, the SEU rate used is the product of the acceleration factor and the normalized λ_{S} value used for the FMF*. As it can be seen from Fig. 4, the quotient is equal to 1 until an acceleration factor bigger than 10^{10}. This means that the periodic derivation FMF* has the same MTTF value than the real FMF function until an acceleration factor of 10^{10}.
3) The TTF histogram can be generated with an acceleration factor of 10^{10}. This accelerated MTTF test can be performed by a software simulator [9], [10], but the amount of simulation time required can be in the order of months for a moderately complex routine. Hence, it is necessary to use a real time or quasi real time fault injection-failure detection scheme like the one discussed in [11]. For the program under study in this paper, the histogram obtained for an acceleration factor of 10^{10} is shown in Fig. 5.

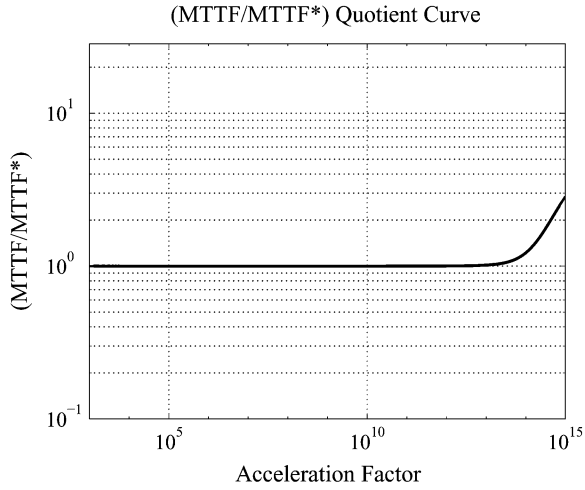

Fig. 4. The (MTTF/MTTF*) quotient curve. Note the flat zone used to determine the maximum acceleration factor.

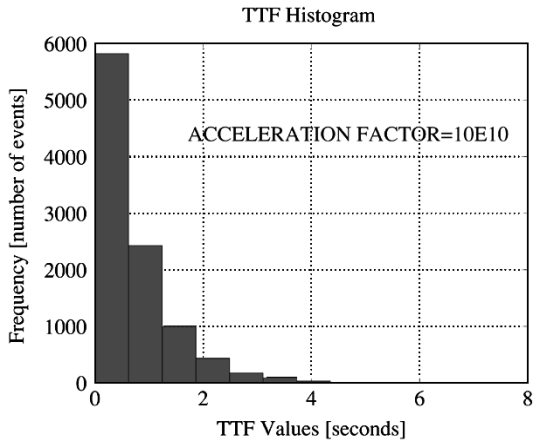

Fig. 5. TTF histogram obtained by means of a real time fault injection-failure detection scheme with an acceleration factor of 10^{10}.

Once the TTF histogram is obtained with an accelerated mean time to failure test, the real TTF values are calculated with a scaling procedure. All the classical statistical data analysis can be performed to fit a convenient model to the system TTF behavior. Finally, the application cross section and the application failure rate can be derived from the information generated in the above procedures.

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.

VII. Conclusions and Future Work

The TTF probability density functions, and/or the cumulative density function contain much more information of the system's behavior than the application cross section σ_{AP}. In fact, σ_{AP} is only one type of the information that can be obtained from these functions. Also, the tools presented are a powerful way to visualize the system's fault tolerance and reliability properties by means of graphical representation, such as the failure map, reliability and availability curves, etc. Failure Maps allow the classification of applications in two main groups: periodic and nonperiodic. A periodic application is very easy to handle as shown previously with the FMF based approach. Non periodic applications are in general more difficult to handle but in those cases a periodic derivation FMF* function can be constructed to calculate the largest SEU rate, where the MTTF is related to the application cross section in a linear way. Then an accelerated MTTF test can be performed reducing the simulation time several orders of magnitude. A wide range of applications can be analyzed by means of these methods, including operating systems, and combination of random sequences of complex routines. This represents, to our knowledge a very important contribution to the state of the art in the field, since a method to obtain σ_{AP} in such applications has not been clearly established yet. Moreover, since σ_{AP} and the system's MTTF are related, the experimental methods (radiation tests) used to validate the σ_{AP} values, can also be used to validate these two new methods. One of the next steps in this work will be the comparison of data obtained from simulation and radiation testing.

The main limitation of this work is the *simplifying assumption* that all internal sensitive nodes (bits) have the same cross-section, and that this value is known. Hence, the results obtained are only an approximation. However, (7) remains a valid expression for $\lambda_F(t)$ if the FMF(t) is defined as in (10) or (11), and the rest of the procedure remains unchanged. As stated before, the values $\sigma_{\mathrm{i}}\ \mathrm{i} = 1, \ldots, \mathrm{N}$ can be considered random variables with mean value $(\sigma_{\mathrm{P}}/\mathrm{N})$. This implies that if the number of relevant bits $(\mathrm{N_{re}})$ is large, (11) is a very good approximation for (10). In the case that $\mathrm{N_{re}}$ is a small number, (10) gives an exact value for $\mathrm{FMF}(t)$. Hence, the method admits a generalization to get into consideration the different cross section values of the individual sensitive nodes. Moreover, since it is very difficult to obtain the experimental values for each sensitive node in the device, a Monte Carlo simulation is desirable to analyze the system's behavior with stochastic changes in the σ_i values.

This work will continue with the application of the method to more complex routines and Monte Carlo simulations to analyze the impact of changes in the sensitive node cross section values.

Another limitation of the proposed method is the need of the processor's internal circuitry information to obtain the $\mathrm{FMF_i(t)}$ functions. However, the software tools developed in this work give suitable approximations for the $\mathrm{FMF_i(t)}$ functions. In particular, the RTL model approach described in Section IV, gives the more accurate results when the processor information circuitry is not available. This work will continue with a comparative study of the different software tools developed to obtain the $\mathrm{FMF_i(t)}$ functions.

Finally, it is important to remark that the effort in the development of the required fault injection failure detection tools is similar to the effort required in the development of the tools used in the application cross section approach described in [7], and the tools described in [9]–[11].

Acknowledgment

The authors would like to express their special acknowledgment for the technical assistance and advise of Dr. R. Velazco and his Qualification Group at TIMA, Grenoble, France.

References

[1] R. Koga, W. A. Kolasinsky, and S. Imamoto, "Heavy ion induced upsets in semiconductor devices," *IEEE Trans. Nucl. Sci.*, vol. NS-32, no. 1, Feb. 1985.

[2] J. H. Elder, J. Osborn, W. A. Kolasinsky, and R. Koga, "A method for characterizing a microprocessor's vulnerability to SEU," *IEEE Trans. Nucl. Sci.*, vol. 35, no. 6, pp. 1678–1681, Dec. 1988.

[3] R. H. Sorensen *et al.*, "SEU risk assessment of Z80A, 8086, and 80C86 microprocessors intended for use in a low altitude polar orbit," *IEEE Trans. Nucl. Sci.*, vol. NS-33, 1986.

[4] R. Koga *et al.*, "Techniques of microprocessor testing and SEU rate prediction," *IEEE Trans. Nucl. Sci.*, vol. NS-32, 1985.

[5] F. Bezerra, R. Velazco, A. Assoum, and D. Benezech, "SEU and latch up results on transputers," *IEEE Trans. Nucl. Sci.*, vol. 43, no. 3, June 1996.

[6] S. H. Crain, R. Velazco, A. Bofill, P. Yu, and R. Koga, "Radiation effects in a fixed-point digital signal processor," in *Proc. IEEE Nuclear and Space Radiation Effects Data Workshop (NSREC 99)*, Norfolk, VA, July, 12–16 1999.

[7] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodriguez, and J. Mingo, "Estimating error rates in processor-based architectures," *IEEE Transactions on Nuclear Science*, vol. 48, no. 5, Oct. 2001.

[8] R. Velazco, Ph. Cheynet, A. Bofill, and R. Ecoffet, "THESIC: A testbed suitable for the qualification for integrated circuits devoted to operate in harsh environment," in *IEEE European Test Workshop (ETW98)*, Barcelona, Spain, May, 27–29 1998.

[9] P. A. Ferreyra, C. A. Marqués, J. P. Gaspar, and R. T. Ferreyra, "A software tool for simulating single event upsets in a digital signal processor," in *2nd IEEE Latin-American Test Workshop*, Cancún, México, Feb. 11–14, 2001.

[10] R. Velazco, A. Corominas, and P. A. Ferreyra, "Injecting bit flips by means of a purely software approach: A case studied," *Defect and Fault Tolerance in VLSI Systems*, Nov. 5–7, 2002.

[11] P. A. Ferreyra, C. A. Marqués, R. Velazco, and O. Calvo, "Injecting single event upsets in a digital signal processor by means of direct memory access requests: A new method for generating bit flips," in *Proc. RADECS 2001, (Radiation and Its Effects on Components and Systems)*, Grenoble, France, Sep. 10–14, 2001.

[12] A. Papoulis, *Probability, Random Variables, and Stochastic Process*, 3rd ed: McGraw-Hill International Editions, 1991.

Authorized licensed use limited to: Intel Corporation via the Intel Library. Downloaded on February 27, 2009 at 14:59 from IEEE Xplore. Restrictions apply.