SER within the prescribed budget. Often this process is iterated several times till designers are happy with the predicted SER. This book describes the current state-of-the-art in soft error modeling, measurement, detection, and correction mechanisms.

This chapter reviews basic definitions of faults, errors, and metrics, and dependability models. Then, it shows how these definitions and metrics apply to both permanent and transient faults. The discussion on permanent faults will place radiation-induced transient faults in a broader context, covering various silicon reliability problems.

1.2 Faults

User-visible *errors,* such as soft errors, are a manifestation of underlying *faults* in a computer system. Faults in hardware structures or software modules could arise from defects, imperfections, or interactions with the external environment. Examples of faults include manufacturing defects in a silicon chip, software bugs, or bit flips caused by cosmic ray strikes.

Typically, faults are classified into three broad categories—permanent, intermittent, and transient. The names of the faults reflect their nature. Permanent faults remain for indefinite periods till corrective action is taken. Oxide wearout, which can lead to a transistor malfunction in a silicon chip, is an example of a permanent fault. Intermittent faults appear, disappear, and then reappear and are often early indicators of impending permanent faults. Partial oxide wearout may cause intermittent faults initially. Finally, transient faults are those that appear and disappear. Bit flips or gate malfunction from an alpha particle or a neutron strike is an example of a transient fault and is the subject of this book.

Faults in a computer system can occur directly in a user application, thereby eventually giving rise to a user-visible error. Alternatively, it can appear in any abstraction layer underneath the user application. In a computer system, the abstraction layers can be classified into six broad categories (Figure 1.2)—user application, OS, firmware, architecture, circuits, and process technology. Software bugs are faults arising in applications, OSs, or firmware. Design faults can arise in architecture or circuits. Defects, imperfections, or bit flips from particle strikes are examples of faults in the process technology or the underlying silicon chip.

A fault in a particular layer may not show up as a user-visible error. This is because of two reasons. First, a fault may be masked in an intermediate layer. A defective transistor—perhaps arising from oxide wearout—may affect performance but may not affect correct operation of an architecture. This could happen, for example, if the transistor is part of a branch predictor. Modern architectures typically use a branch predictor to accelerate performance but have the ability to recover from a branch misprediction.

Second, any of the layers may be partially or fully designed to tolerate faults. For example, special circuits—radiation-hardened cells—can detect and recover

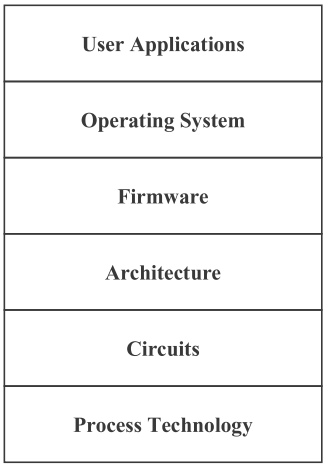

FIGURE 1.2 Abstraction layers in a computer system.

from faults in transistors. Similarly, each abstraction layer, shown in Figure 1.2, can be designed to tolerate faults arising in lower layers. If a fault is tolerated at a particular layer, then the fault is avoided at the layer above it.

The next section discusses how faults are related to errors.

1.3 Errors

Errors are manifestation of *faults*. Faults are necessary to cause an error, but not all faults show up as errors. Figure 1.3 shows that a fault within a particular scope may not show up as an error outside the scope if the fault is either masked or tolerated. The notion of an error (and units to characterize or measure it) is fundamentally tied to the notion of a *scope*. When a fault is detected in a specific scope, it becomes an error in that scope. Similarly, when an error is corrected in a given a scope, its effect usually does not propagate outside the scope. This book tries to use the terms *fault detection* and *error correction* as consistently as possible. Since an error can propagate and be detected again in a different scope, it is also acceptable to use the term *error detection* (as opposed to fault detection).

Three examples are considered here. The first one is a fault in a branch predictor. No fault in a branch predictor will cause a user-visible error. Hence, there is no scope outside which a branch predictor fault would show up as an error. In contrast, a fault in a cache cell can potentially lead to a user-visible error. If the cache cell is protected with ECC, then a fault is an error within the scope of the ECC logic. Outside the scope of this logic where our typical observation point would be, the fault gets tolerated and never causes an error. Consider a third scenario in which three complete microprocessors vote on the correct output. If the output of one of the processors is incorrect, then the voting logic assumes that the other two are

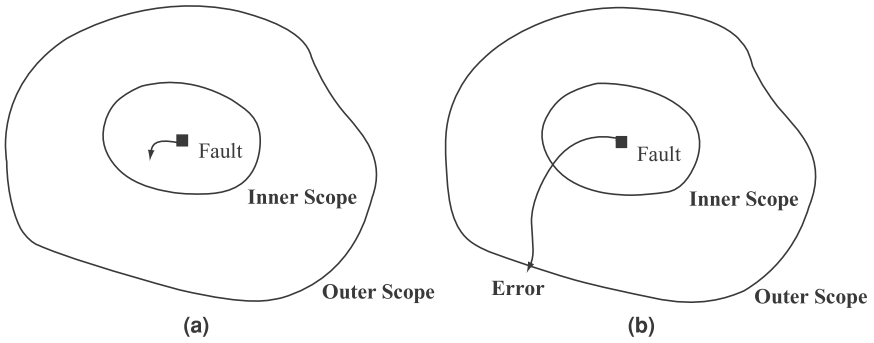

FIGURE 1.3 (a) Fault within the inner scope masked and not visible outside the inner scope. (b) Fault propagated outside the outer scope and visible as an error.

correct, thereby correcting any internal fault. In this case, the scope is the entire microprocessor. A fault within the microprocessor will never show up outside the voting logic.

In traditional fault-tolerance literature, a third term—*failures*—is used besides faults and errors. Failure is defined as a system malfunction that causes the system to not meet its correctness, performance, or other guarantees. A failure is, however, simply a special case of an error showing up at a boundary where it becomes visible to the user. This could be an SDC event, such as a change in the bank account, which the user sees. This could also be a detected error (or DUE) caught by the system but not corrected and may lead to temporary unavailability of the system itself. For example, an ATM machine could be unavailable temporarily due to a system reboot caused by a radiation-induced bit flip in the hardware. Alternatively, a disk could be considered to have failed if its performance degrades by 1000x, even if it continues to return correct data.

Like faults, errors can be classified as permanent, intermittent, or transient. As the names indicate, a permanent fault causes a permanent or hard error, an intermittent fault causes an intermittent error, and a transient fault causes a transient or soft error. Hard errors can cause both infant mortality and lifetime reliability problems and are typically characterized by the classic bathtub curve, shown in Figure 1.4. Initially, the error rate is typically high because of either bugs in the system or latent hardware defects. Beyond the infant mortality phase, a system typically works properly until the end of its useful lifetime is reached. Then, the wearout accelerates causing significantly higher error rates. The silicon industry typically uses a technique called *burn-in* to move the starting use point of a chip to the beginning of the useful lifetime period shown in Figure 1.4. Burn-in removes any chips that fail initially, thereby leaving parts that can last through the useful lifetime period. Further, the silicon industry designs technology parameters, such as oxide thickness, to guarantee that most chips last a minimal lifetime period.

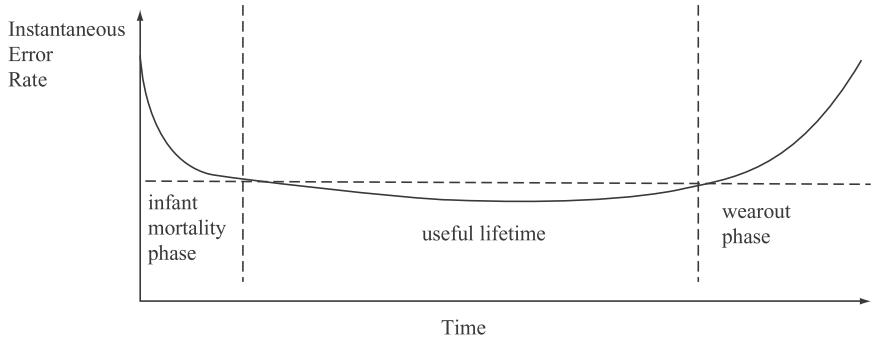

FIGURE 1.4 Bathtub curve showing the relationship between failure rate, infant mortality, useful lifetime, and wearout phase.

1.4 Metrics

Time to failure (TTF) expresses fault and error rates, even though the term TTF refers specifically to failures. As the name suggests, TTF is the time to a fault or an error, as the case may be. For example, if an error occurs after 3 years of operation, then the TTF of that system for that instance is 3 years. Similarly, MTTF expresses the mean time elapsed between two faults or errors. Thus, if a system gets an error every 3 years, then that system's MTTF is 3 years. Sometimes reliability models use median time to failure (MeTTF), instead of MTTF, such as in Black's equation for electromigration (EM)–related errors (see Electromigration, p. 15).

Under certain assumptions (e.g., an exponential TTF, see Reliability, p. 12), the MTTF of various components comprising a system can be combined to obtain the MTTF of the whole system. For example, if a system is composed of two components, each with an MTTF of 6 years, then the MTTF of the whole system is

$$\mathrm{MTTF}_{\mathrm{system}} = \frac{1}{\frac{1}{\mathrm{MTTF}_{\mathrm{component\,1}}} + \frac{1}{\mathrm{MTTF}_{\mathrm{component\,2}}}} = \frac{1}{\frac{1}{6} + \frac{1}{6}} = 3$$

More generally,

$$\mathrm{MTTF}_{\mathrm{system}} = \frac{1}{\sum_{i=0}^{n} \frac{1}{\mathrm{MTTF}_i}}$$

Although the term MTTF is fairly easy to understand, computing the MTTF of a whole system from individual component MTTFs is a little cumbersome, as expressed by the above equations. Hence, engineers often prefer the term *failure in time* (FIT), which is additive.

One FIT represents an error in a billion (10^9) hours. Thus, if a system is composed of two components, each having an error rate of 10 FIT, then the system has a total error rate of 20 FIT. The summation assumes that the errors in each component are independent.

The error rate of a component or a system is often referred to as its FIT rate. Thus, the FIT rate equation of a system is

$$\text{FIT rate}_{\text{system}} = \sum_{i=0}^{n} \text{FIT rate}_i$$

As may be evident by now, FIT rate and MTTF of a component are inversely related under certain conditions (e.g., exponentially distributed TTF):

$$\text{MTTF in years} = \frac{10^9}{\text{FIT rate} \times 24 \text{ hours} \times 365 \text{ days}}$$

Thus, an MTTF of 1000 years roughly translates into a FIT rate of 114 FIT.

EXAMPLE

A silicon chip consists of a billion transistors, each with a FIT rate of 0.00001 FIT. What will be the MTTF of a system composed of 100 such chips?

SOLUTION The FIT rate of each chip = $10^9 \times 0.00001$ FIT = 10^4 FIT. The FIT rate of 100 such chips = $100 \times 10^4 = 10^6$ FIT. Then, the MTTF of a system with 100 such chips = $10^9/(10^6 \times 24) \sim 40$ days.

EXAMPLE

What is the MTTF of a computer's memory system that has 16 gigabytes of memory? Assume FIT per bit is 0.00001 FIT.

SOLUTION The FIT rate of the memory system = $16 \times 2^{30} \times 8 \times 0.00001 = 1\,374\,390$ FIT. This translates into an MTTF of $10^9/(1\,374\,390 \times 24) \sim 30$ days.

Besides MTTF, two terms—*mean time to repair* (MTTR) and *mean time between failures* (MTBF)—are commonly used in the fault-tolerance literature. MTTR represents the mean time needed to repair an error once it is detected. MTBF

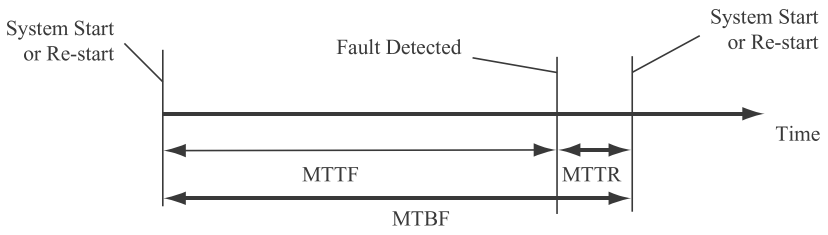

FIGURE 1.5 Relationship between MTTF, MTTR, and MTBF.

represents the average time between the occurrences of two errors. As Figure 1.5 shows, MTBF = MTTF + MTTR. Typically, MTTR $\ll$ MTTF. The next section examines how these terms are used to express various concepts in reliable computing.

Recently, Weaver et al. [26] introduced the term *mean instructions to failure* (MITF). MITF captures the average number of instructions committed in a microprocessor between two errors. Similarly, Reis et al. [19] introduced the term *mean work to failure* (MWTF) to capture the average amount of work between two errors. The latter is useful for comparing the reliability for different workloads. Unlike MTTF, both MITF and MWTF try to capture the amount of work done till an error is experienced. Hence, MITF and MWTF are often useful in doing trade-off studies between performance and error rate.

The definitions of MTTF and FIT rate have one subtlety that may not be obvious. Both terms are related to a particular scope (as explained in the last section). Consider a bit with ECC, which can correct an error in the single bit. The MTTF(bit) is significantly lower than the MTTF(bit + ECC). Conversely, the FIT rate(bit) is significantly greater than the FIT rate(bit + ECC). In both cases, it is the MTTF that is affected and not the MTBF. Vendors, however, sometimes incorrectly report MTBF numbers for the components they are selling, if they add error correction to the component.

All the above metrics can be applied separately for SDC or DUE. Thus, one can talk about SDC MTTF or SDC FIT. Similarly, one can express DUE MTTF or DUE FIT. Usually, the total SER is expressed as the sum of SDC FIT and DUE FIT.

1.5 Dependability Models

Reliability and availability are two attributes typically used to characterize the behavior of a system experiencing faults. This section discusses mathematical models to describe these attributes and the foundation behind the metrics discussed in the last section. This section will also discuss other miscellaneous related models used to characterize systems experiencing faults.

1.5.1 Reliability

The reliability $R(t)$ of a system is the probability that the system does not experience a user-visible error in the time interval $(0, t]$. In other words, $R(t) = P(T > t)$, where T is a random variable denoting the lifetime of a system. If a population of N_0 similar systems is considered, then $R(t)$ is the fraction of the systems that survive beyond time t. If N_t is the number of systems that have survived until time t and $E(t)$ is the number of systems that experienced errors in the interval $(0, t]$, then

$$R(t) = \frac{N_t}{N_0} = \frac{N_0 - E(t)}{N_0} = 1 - \frac{E(t)}{N_0}$$

Differentiating this equation, one gets

$$\frac{dR(t)}{dt} = -\frac{\frac{dE(t)}{dt}}{N_0}$$

The instantaneous error rate or hazard rate $h(t)$—graphed in Figure 1.4—is defined as the probability that a system experiences an error in the time interval Δt, given that it has survived till time t. Intuitively, $h(t)$ is the probability of an error in the time interval $(t, t + \Delta t]$.

$$h(t) = P(t < T \leq t + \Delta t | (T > t)) = \frac{\frac{dE(t)}{dt}}{N_t} = \frac{\frac{\frac{dE(t)}{dt}}{N_0}}{\frac{N_t}{N_0}} = \frac{\frac{dR(t)}{dt}}{R(t)}$$

Rewriting,

$$\frac{dR(t)}{dt} = -h(t)R(t)$$

The general solution to this differential equation is

$$R(t) = e^{-\int h(t)dt}$$

If one assumes that $h(t)$ has a constant value of λ (e.g., during the useful lifetime phase in Figure 1.4), then

$$R(t) = e^{-\lambda t}$$

This exponential relationship between reliability and time is known as the *exponential failure law,* which is commonly used in soft error analysis. The expectation of $R(t)$ is the MTTF and is equal to λ.

The exponential failure law lets one sum FIT rates of individual transistors or bits in a silicon chip. If it is assumed that a chip has n bits, where the ith bit has a constant and independent hazard rate of h_i, then, $R(t)$ of the whole chip can be expressed as

$$R(t) = \prod_{i=0}^{n-1} R_i(t) = \prod_{i=0}^{n-1} e^{-h_i t} = e^{-\left(\sum_{i=0} h_i\right) t}$$

Thus, the reliability function of the chip is also exponentially distributed with a constant FIT rate, which is the sum of the FIT rates of individual bits.

The exponential failure law is extremely important for soft error analysis because it allows one to compute the FIT rate of a system by summing the FIT rates of individual components in the system. The exponential failure law requires that the instantaneous SER in a given period of time is constant. This assumption is reasonable for soft error analysis because alpha particles and neutrons introduce faults in random bits in computer chips. However, not all errors follow the exponential failure law (e.g., wearout in Figure 1.4). The Weibull or log-normal distributions could be used in cases that have a time-varying failure rate function [18].

1.5.2 Availability

Availability is the probability that a system is functioning correctly at a particular instant of time. Unlike reliability, which is defined over a time interval, availability is defined at an instant of time. Availability is also commonly expressed as

$$\text{Availability} = \frac{\text{system uptime}}{\text{system uptime} + \text{system downtime}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} = \frac{\text{MTTF}}{\text{MTBF}}$$

Thus, availability can be increased either by increasing MTTF or by decreasing MTTR.

Often, the term five 9s or six 9s is used to describe the availability of a system. The term five 9s indicates that a system is available 99.999% of the time, which translates to a downtime of about 5 minutes per year. Similarly, the term six 9s indicates that a system is available 99.9999% of the time, which denotes a system downtime of about 32 seconds per year. In general, n 9s indicate two 9s before the decimal point and $(n - 2)$ 9s after the decimal point, if expressed in percentage.

EXAMPLE

If the MTTR of a system is 30 minutes, how many crashes can it sustain per year and still maintain a five 9s uptime? What is the MTTF in this case?

SOLUTION A five 9s uptime denotes a total downtime of about 5 hours per year. Hence, the number of system crashes allowed for this system per year is $(5 \times 60/30) = 10$. The MTTF is (1 year–5 hours)/10 = 876 hours.

1.5.3 Miscellaneous Models

Three other models, namely maitainability, safety, and performability, are often used to describe systems experiencing faults. Maintainability is the probability that a failed system will be restored to its original functioning state within a specific

period of time. Maintainability can be modeled as an exponential repair law, a concept very similar to the exponential failure law.

Safety is the probability that a system will either function correctly or fail in a "safe" manner that causes no harm to other related systems. Thus, unlike reliability, safety modeling incorporates a "fail-stop" behavior. Fail-stop implies that when a fault occurs, the system stops operating, thereby preventing the effect of the fault to propagate any further.

Finally, performability of a system is the probability that the system will perform at or above some performance level at a specific point of time [10]. Unlike reliability, which relates to correct functionality of all components, performability measures the probability that a subset of functions will be performed correctly. Graceful degradation, which is a system's ability to perform at a lower level of performance in the face of faults, can be expressed in terms of a performability measure.

These models are added here for completeness and will not be used in the rest of this book. The next few sections discuss how the reliability and availability models apply to both permanent and transient faults.

1.6 Permanent Faults in Complementary Metal Oxide Semiconductor Technology

Dependability models, such as reliability and availability, can characterize both permanent and transient faults. This section examines several types of permanent faults experienced by complementary metal oxide semiconductor (CMOS) transistors. The next section discusses transient fault models for CMOS transistors. This section reviews basic types of permanent faults to give the reader a broad understanding of the current silicon reliability problems, although radiation-induced transient faults are the focus of this book.

Permanent faults in CMOS devices can be classified as either extrinsic or intrinsic faults. Extrinsic faults are caused by manufacturing defects, such as contaminants in silicon devices. Extrinsic faults result in infant mortality, and the fault rate usually decreases over time (Figure 1.4). Typically, a process called burn-in, in which silicon chips are tested at elevated temperatures and voltages, is used to accelerate the manifestation of extrinsic faults. The defect rate is expressed in defective parts per million.

In contrast, intrinsic faults arise from wearout of materials, such as silicon dioxide, used in making CMOS transistors. In Figure 1.4, the intrinsic fault rate corresponds to the wearout phase and typically increases with time. Several architecture researchers are examining how to extend the useful lifetime of a transistor device by delaying the onset of the wearout phase and decreasing the use of the device itself.