Gil Tomás, DA.; Gracia-Morán, J.; Baraza Calvo, JC.; Saiz-Adalid, L.; Gil Vicente, PJ. (2012). Studying the effects of intermittent faults on a microcontroller. Microelectronics Reliability. 52(11):2837-2846. doi:10.1016/j.microrel.2012.06.004.

Elsevier Editorial System(tm) for Microelectronics Reliability
Manuscript Draft

Manuscript Number:

Title: A Comparative Study of the Effects of Intermittent Faults in a Microcontroller

Article Type: Original Research Paper

Keywords: Fault injection; Hardware description languages; Integrated circuit reliability; Intermittent faults; Microcontrollers

Corresponding Author: Dr. Joaquin Gracia-Moran,

Corresponding Author's Institution: Grupo de Sistemas Tolerantes a Fallos (GSTF), Instituto ITACA, Universitat Politècnica de València

First Author: Daniel Gil-Tomas

Order of Authors: Daniel Gil-Tomas; Joaquin Gracia-Moran; J.-Carlos Baraza; Luis J Saiz-Adalid; Pedro J Gil-Vicente

Abstract: As CMOS technology scales to the nanometer range, designers have to deal with a growing number and variety of fault types. Particularly, intermittent faults are expected to be an important issue in modern VLSI circuits. The complexity of manufacturing processes, producing residues and parameter variations, together with special aging mechanisms, may increase the presence of such faults. This work presents a case study of the impact of intermittent faults on the behavior of a commercial microcontroller. In order to carry out an exhaustive reliability assessment, the methodology used lies in VHDL-based fault injection techniques. In this way, a set of intermittent fault models at logic and register transfer abstraction levels have been generated and injected in the VHDL model of the system. From the simulation traces, the occurrences of failures and latent errors have been logged. The impact of intermittent faults has been also compared to that got when injecting transient and permanent faults. The results obtained in this work suggest the suitability of adding mitigation techniques to deliver fast error detection and correction of intermittent faults in buses and critical registers.

A Comparative Study of the Effects of Intermittent Faults in a Microcontroller

Daniel Gil-Tomás, Joaquín Gracia-Morán, J.-Carlos Baraza-Calvo, Luis-J. Saiz-Adalid, Pedro-J. Gil-Vicente
All authors are with the Grupo de Sistemas Tolerantes a Fallos (GSTF), Instituto ITACA, Universitat Politècnica de València.
Postal address:
Escuela Técnica Superior de Ingeniería Informática, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain.
Corresponding author:
Joaquín Gracia-Morán, phone: +34963875722; fax: +34963877579; email: jgracia@disca.upv.es, postal address: Escuela Técnica Superior de Ingeniería Informática, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain.
e-mail authors: {dgil, jgracia, jcbaraza, ljsaiz, pgil}@disca.upv.es

Abstract
As CMOS technology scales to the nanometer range, designers have to deal with a growing number and variety of fault types. Particularly, intermittent faults are expected to be an important issue in modern VLSI circuits. The complexity of manufacturing processes, producing residues and parameter variations, together with special aging mechanisms, may increase the presence of such faults. This work presents a case study of the impact of intermittent faults on the behavior of a commercial microcontroller. In order to carry out an exhaustive reliability assessment, the methodology used lies in VHDL-based fault injection techniques. In this way, a set of intermittent fault models at logic and register transfer abstraction levels have been generated and injected in the VHDL model of the system. From the simulation traces, the occurrences of failures and latent errors have been logged. The impact of intermittent faults has been also compared to that got when injecting transient and permanent faults. The results obtained in this work suggest the suitability of adding mitigation techniques to deliver fast error detection and correction of intermittent faults in buses and critical registers.

Keywords
Fault injection; Hardware description languages; Integrated circuit reliability; Intermittent faults; Microcontrollers

1. Introduction

As technology has progressed toward nanoscale devices, microprocessors and microcontrollers have reduced their area and supply voltage, increasing its processing speed. However, these advances impact negatively on their reliability. As feature sizes are aggressively scaled, the manufacturing of integrated circuits becomes more complex so that more defects and process variations are introduced. In addition, deep submicron devices are more vulnerable to the environment due to the reduction of the energy required to enable switching. As a result, designers have to deal with a growing number and variety of fault types [1].
Usually, only the effects of permanent and transient faults have been studied and modeled [2, 3]. Permanent faults, also known as hard faults, are caused by irreversible physical changes in a chip. Although the most common sources for this kind of faults are manufacturing processes, they can occur also during the operation of the circuit, especially when the circuit is old and starts to wear out.
On the other hand, transient faults are commonly generated by temporary environmental conditions, like electromagnetic interferences or cosmic radiation. This type of faults does not leave any permanent effects on the chip and thus they are also called soft errors. The occurrence of transient faults is commonly random and, therefore, hard to detect.
However, intermittent faults have not been considered habitually. These faults manifest as occasional

bursts that typically repeat themselves every now and then, and whose effects are not continuous as permanent faults. Also, intermittent faults occur repeatedly in the same places, and they may be activated or deactivated by changes in temperature, voltage or frequency [4]. Nowadays, intermittent faults are expected to have a great impact in nanotechnologies. The complexity of the manufacturing process (which produces residues and process variations) and special aging mechanisms may increase the presence of intermittent faults [1]. In addition, errors provoked by intermittent faults are very hard to detect because they may only occur under certain environmental constraints or in the presence of some specific input vector combination [5].

Fault Injection is a common method to assess the reliability of a computer system [6]. This technique allows a controlled introduction of faults in the system, not being necessary to wait for a long time to log the apparition of real faults. Fault injection techniques can be classified in three main categories [3]: physical (or Hardware Implemented Fault Injection, HWIFI), software implemented (SWIFI) and simulation-based.

Simulation-based fault injection is a useful experimental way to evaluate the dependability of a system during the design phase. An early diagnosis allows saving costs in the design process, avoiding redesigning in case of error, and thus reducing time-to-market. In particular, VHDL-based Fault Injection has been used due to its flexibility, as well as the high observability and controllability of all the modeled components [7].

So far, very few attempts have been done in order to study the effects of intermittent faults by fault injection. In most of the works issued, real systems were monitored, and faults and failures produced were observed to determine the most frequent sources of errors and their manifestation [4, 8, 9].

The objective of this work is to study the impact of intermittent faults in a commercial microcontroller, as well as to compare their consequences to those provoked by permanent and transient faults. To do this, a set of intermittent fault models at logic and register transfer (RT) abstraction levels has been generated, and they have been injected in the VHDL model of a commercial microcontroller in order to analyze their impact.

To sum up, the main contributions of this work are:

a. To generate intermittent fault models at logic and register transfer (RT) abstraction levels, paying special attention to faults in combinational logic.
b. To inject these fault models, applying different VHDL-based fault injection techniques.
c. An exhaustive variation of the fault and system parameters, in order to obtain an extensive analysis of their effects.
d. A study of the impact of intermittent faults in the behavior of a commercial microcontroller.
e. Comparing the influence of intermittent faults to that of transient and permanent faults.

The paper is organized as follows. Section 2 describes the generation of intermittent fault models. Section 3 depicts the fault injection experiments. Section 4 includes a selection of the results. Finally, Section 5 provides some conclusions.

2. Intermittent fault models

Whereas transient and permanent fault models have been traditionally well established, intermittent fault modeling is a pending issue [1]. Examples of popular fault models for permanent and transient faults are *stuck-at* and *bit-flip*, respectively [3].

To obtain representative fault models for intermittent faults, it is necessary to understand the physical mechanisms that take place in deep submicron technologies. Intermittent faults occur due to unstable or marginal hardware, and they may be activated by an environmental change such as temperature or voltage alterations. Manufacturing residues, process variations and special aging processes can lead to such faults. Traditionally, permanent fault models have been applied to intermittent faults, as intermittent faults often precede the occurrence of a permanent fault. Nevertheless, the introduction of new submicron technologies makes necessary to study new fault causes and mechanisms of intermittent faults. In this way, Table 1 summarizes some representative physical causes and fault mechanisms of intermittent faults, as well as the fault models proposed in every case.

To do that we have selected a set of intermittent faults observed in real computer systems by means of fault logging [4, 9], and analyzed some fault mechanisms related to process variations and wearout that can provoke intermittent faults [10-15]. Then, a set of fault models at logic and RT abstraction levels which can be simulated into VHDL models has been deduced.

The table tries to unify, classify and relate the different fault sources. It shows intermittent fault models

for buses, storage elements, input/output connections and combinational logic. It extends the study made in [16] and [17], adding new challenging wearout processes of the nano-CMOS technology.

TABLE 1. Some intermittent fault mechanisms and models

Causes	Targets	Fault mechanisms	Type of fault	Fault models
Residues in cells	Memory and registers	Intermittent contacts	Manufacturing defect	*Intermittent stuck-at*
Solder joints	Buses	Intermittent contacts	Manufacturing defect	*Intermittent pulse* *Intermittent short* *Intermittent open*
Electromigration Delamination	Buses I/O connections	Variation of metal resistance Voids	Wearout-Timing	*Intermittent delay* *Intermittent short* *Intermittent open*
Crosstalk	I/O connections Buses	Electromagnetic interference	Internal noise Timing	*Intermittent pulse* *Intermittent delay* *Intermittent speed-up*
Gate oxide soft breakdown	NMOS transistors in SRAM cells	Leakage current fluctuation	Wearout-Timing	*Intermittent delay* *Intermittent Indetermination*
Negative bias-temperature instability (NBTI)	PMOS transistors in combinational logic	Increase of transistor threshold voltage V_{TH} Reduction of carrier mobility	Wearout-Timing	*Intermittent delay*
Negative bias-temperature instability (NBTI)	PMOS transistors in SRAM cells	Local mismatches among cell transistors, decrease of static noise margin	Wearout	*Intermittent bit-flip*
Hot-carrier injection (HCI)	NMOS transistors in combinational logic	Increase of transistor threshold voltage V_{TH}	Wearout-Timing	*Intermittent delay*
Low-k dielectric breakdown	Buses I/O connections	Leakage current fluctuation Temperature variations Capacity degradation	Wearout-Timing	*Intermittent delay* *Intermittent short*
Doping profile and gate length deviations	MOS transistors in combinational logic and memory	Deviations in V_{TH} Deviations in operation speed	Manufacturing variations	*Intermittent delay*

According to *type of fault* column, some faults are due to manufacturing defects and residues, which provoke intermittent contacts.

Other faults are caused by wearout or aging mechanisms, affecting the metal connections (electromigration, delamination) or the gate-oxide of the transistors (soft breakdown). These may produce intermittent contacts and timing errors provoked by resistance variations and leakage current fluctuations. Timing errors and signal perturbations can also be produced by crosstalk between adjacent signals.

In order to extend the fault models to combinational logic, two cases have been considered. Firstly, faults that affect the input/output connections of the combinational logic; and secondly, faults in internal components. In this last case, some challenging wearout processes that affect the gate-oxide and the interconnect-oxide have been considered: *negative bias-temperature instability (NBTI)*, *hot-carrier injection (HCI)* and *low-k dielectric breakdown*.

NBTI is a PMOS-specific transistor-aging effect that increases the threshold voltage (V_{TH}) of a device, and reduces the carrier mobility μ as a function of time and stress condition [12]. NBTI can result from continuous trap generation in a transistor's Si-SiO_2 interface. These traps usually originate from Si-H bonds generated after the hydrogen passivation process to remove dangling silicon atoms at the Si-SiO_2 interface. However, under stressed operating conditions (negative gate bias at high temperatures), these bonds can easily break with time and generate positive interfacial traps, which increase V_{TH}. Although the device reliability community has been aware of the NBTI process for decades, NBTI has recently gained more attention, mainly because of the wide use of ultrathin oxide devices. These thin oxides substantially increase the vertical oxide field and provoke more severe degradations. The increase of the PMOS transistor threshold voltage can lead, in his turn, to the increase of circuit delay in random-logic circuits and to the decrease of the static noise margins in memory arrays [14]. The increase of V_{TH} can eventually derive to the transistor *stuck-off*, where the transistor is permanently in non conducting state.

HCI is also related to the degradation of the gate oxide. As electrons are accelerated along the channel, they can acquire enough energy so that, through scattering and/or impact ionization, they can be injected into the gate oxide causing interface-state generation [12]. The effects are similar to those of NBTI, but in NMOS transistors.

Besides transistors, scaled interconnections also suffer the rapid increase of reliability problems, especially after the introduction of new materials (such as copper or low-k dielectrics). These new materials enhance wire performance, but degrade thermal and mechanical stability. The situation is compounded by variations in the line geometry that increase the failure probability. One of the emerging

wearout issues is low-k dielectric breakdown [15]. In recent years, copper and low-k interconnect systems have become vulnerable to breakdown because of the lower breakdown field strengths of porous low-k materials, the susceptibility of low-k materials to mechanical damage by chemical mechanical polishing, and the high susceptibility of low-k materials to copper drift. Low-k dielectric breakdown provokes fluctuations of the leakage currents and the wire capacity in time, which in his turn can lead to the increase of circuit delay in random-logic circuits. Dielectric breakdown can also produce intermittent shorts that eventually lead to permanent breakdown.
These aging mechanisms may manifest as intermittent faults before deriving in permanent faults. Initially, faults will appear intermittently, depending on specific conditions (e.g., voltage, temperature, circuit inputs, etc.), but they might eventually end up as permanent defects.
In short, the proposed fault models for the combinational logic are: *intermittent {delay, short, open, pulse, stuck-at, indetermination}*.
Finally, another cause of intermittent faults can be the manufacturing variations, as it is shown in Table 1. As the fabrication dimensions shrink, the proportional extent of random deviations becomes larger and their effects more severe. For instance, gate length deviations and fluctuations of the doping profile provoke deviations of the transistor threshold voltage and the speed operation [5]. Slower devices may lead to timing violations and therefore to the malfunction of the circuit. This manifests as intermittent faults because the circuit may correctly operate most of the time.

3. Fault injection experiments

To inject the faults, VFIT (VHDL-based Fault Injection Tool) [3], a tool developed by the GSTF has been used. VFIT is able to inject faults automatically applying *simulator commands*, *saboteurs* and *mutants* techniques [18].
Fig. 1 shows the classification of the different VHDL-Based Fault Injection Techniques [3].

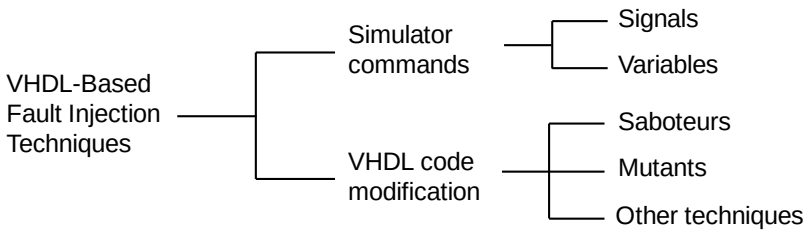


Fig. 1. Fault injection techniques for VHDL models.

Simulator commands change automatically, at simulation time, the value or the timing of the signals and variables of the system. *Saboteurs* and *mutants* techniques are more complex, as they modify the VHDL code of the system by inserting injection components (*saboteurs*) or activating "mutated" versions of the existing components (*mutants*). *Other techniques* are based in some extensions of the syntax and semantics of the VHDL language.
Most of the experiments have been carried out with *simulator commands* technique, because it is not necessary to modify the VHDL code and it is easy to apply. *Saboteurs* technique has been used in the last group of experiments, in order to extend the fault load injected.
Intermittent faults have been injected on the VHDL model of the 8051 microcontroller [19], running the Bubblesort sorting algorithm as workload.
Fig. 2 shows the structure of the 8051 microcontroller and the injection targets of the experiments. Intermittent faults have been injected in the storage elements (the register file and the internal RAM), the internal buses, and the combinational logic of the ALU and CU.

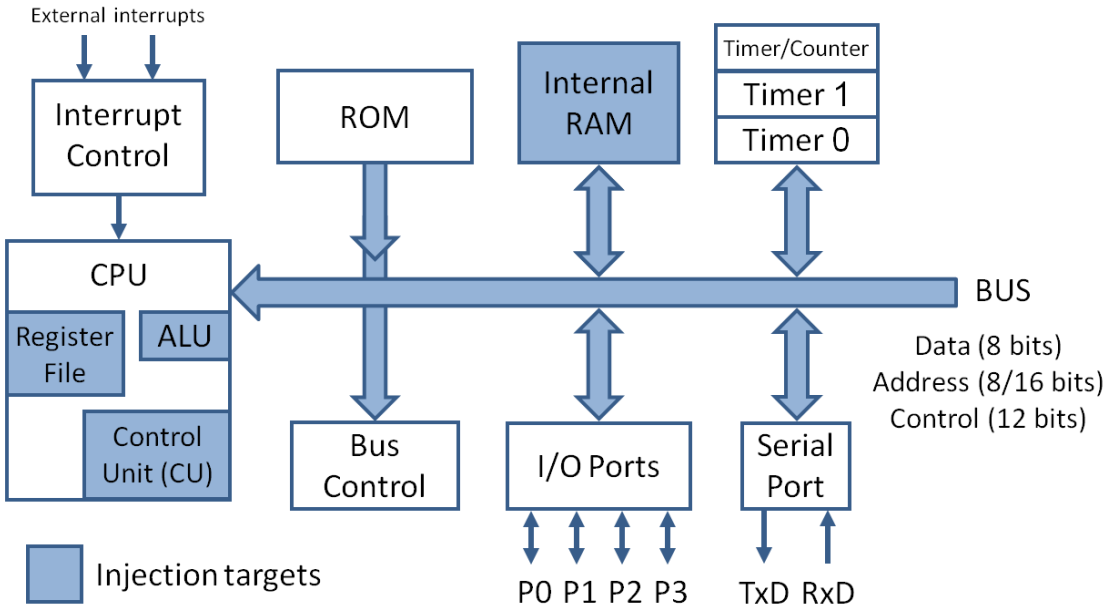
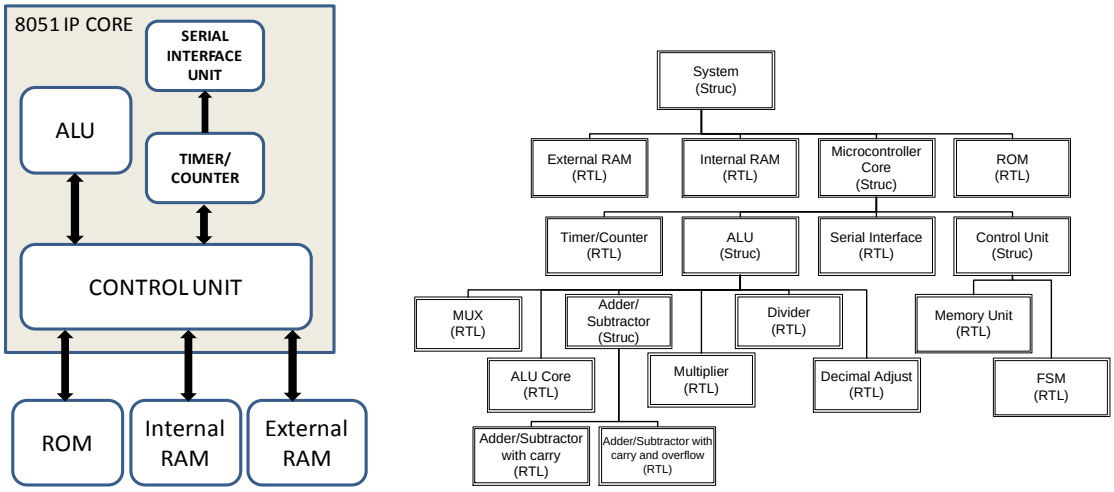


Fig. 2. Structure of the 8051 microcontroller and injection targets.

Fig. 3 shows the block diagram and the design hierarchy of the microcontroller VHDL model. Modules containing logic have been designed with a behavioral RTL architecture, whereas modules composed of connected sub-modules present a structural architecture. The type of description is shown in each module: *Struc* (structural) or *RTL* (behavioral at RT level).
Although the chosen system is a small prototype processor, the methodology used to analyze the impact of intermittent faults would be the same for more complex microcontrollers and microprocessors.



(a) Block diagram. (b) Design hierarchy.

Fig. 3. 8051 microcontroller VHDL model. (a) Block diagram. (b) Design hierarchy.

The main injection parameters are:

a) *Fault multiplicity*

We have injected both single and multiple faults. Due to technology scaling, it is expected that intermittent faults will likely affect multiple locations [9]. Multiple faults have been injected in adjacent and non-adjacent places. To select the targets, Uniform distribution functions have been used.

b) *Fault models*

According to Section 2, and considering the capabilities of *simulator commands* fault injection technique, the intermittent fault models selected have been:

- *Intermittent stuck-at*, for storage elements.
- *Intermittent pulse*, for buses.
- *Intermittent* {*pulse, open, stuck-at, indetermination*}, for combinational logic.

Delay fault model has not been injected with *simulator commands* because of the lack of temporal specifications in the VHDL model of the core. Nevertheless, by using the *saboteurs* technique, the faults injected in the buses have been extended with two new fault models, *Intermittent short* and *Intermittent delay*.

c) *Injection instant*

It has been selected randomly along the workload duration, according to a Uniform distribution.
In real computer systems, other fault distributions have been observed, such as Exponential or Weibull [18]. For instance, a Weibull distribution with increasing fault rate can be used to emulate a wearout process that increases the frequency of intermittent faults, before eventually become permanent.

d) Burst parameters

Intermittent faults manifest in bursts. So, to inject this type of faults the following parameters can be configured [17] (see Fig. 4):

- The number of fault activations in the burst (we will call it *burst length*, or L_{Burst}).
- The duration of each activation (we will refer to it as *activity time*, or t_A).
- The separation between two consecutive activations (we will name it as *inactivity time*, or t_I).

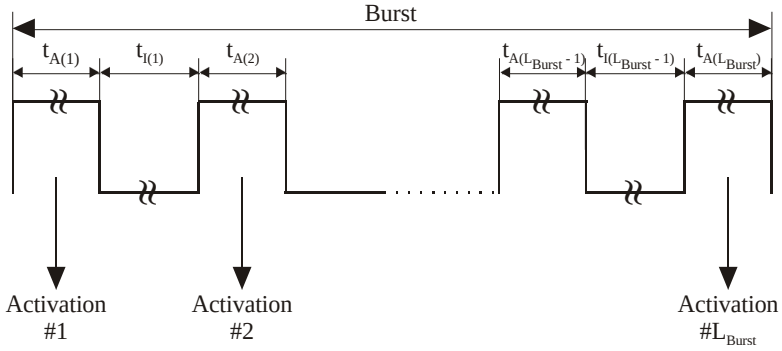


Fig. 4. Main elements of an intermittent burst.

All the three parameters were generated according to random Uniform distribution functions. L_{Burst} was varied between 1 and 10. For t_A and t_I, three time ranges were used: [0.01T–0.1T], [0.1T–1.0T], and [1.0T–10.0T], where T is the clock cycle. In our experiments, the nominal clock frequency of the microcontroller is 10 MHz, thus T = 100 ns.
Intermittent faults which become permanent will probably vary their burst parameters along time, increasing L_{Burst} and t_A, and decreasing t_I. Other distribution functions, like Weibull or Lognormal, may be used to emulate aging processes.

e) Number of faults injected

To obtain a reliable statistic sample, 1,000 faults have been injected per experiment, so that more than 100,000 faults have been injected in total.

f) Measures obtained

In order to measure the impact of intermittent faults, we have calculated in every experiment the percentages of provoked *failures* and *latent errors*, defined as follows:

- Percentage of failures:

$$P_{Failures} = \frac{N_{Failures}}{N_{Injected}} \times 100$$

- Percentage of latent errors:

$$P_{Latent} = \frac{N_{Latent}}{N_{Injected}} \times 100$$

where:

- $N_{Injected}$ is the number of faults injected.
- $N_{Failures}$ is the number of failures. A failure is produced when the result obtained after the execution of the workload is erroneous.

- N_{Latent} is the number of latent errors. A latent error is produced when the injected fault propagates to the storage elements but it does not provoke a failure.

Faults that provoke neither failures nor latent errors are called *Non Effective*. Latent errors and failures are detected by comparing the trace of every faulty simulation with a golden run. Fig. 5 summarizes the syndrome of faults and the calculated data.

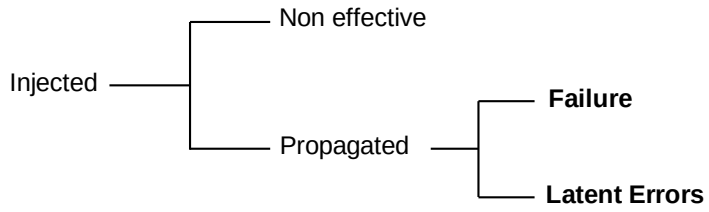


Fig. 5. Syndrome of faults.

4. Results

The results are divided in five groups. Section 4.1 studies the influence of burst parameters. Section 4.2 analyzes the influence of the injection target. Section 4.3 verifies the effect of the system clock frequency. Section 4.4 compares the impact of intermittent faults to that of transient and permanent faults for combinational logic. Finally, Section 4.5 shows the use of the *saboteurs* fault injection technique in order to extend the set of injected fault models.
These results complete those obtained in [16, 17], where faults were injected only in storage elements and buses by applying the *simulator commands* technique.

4.1. Influence of burst parameters

Activity time
Fig. 6 and Fig. 7 represent, respectively, the influence of t_A and the fault multiplicity in the percentages of failures and latent errors. In these figures, t_I has been defined in the intermediate range [0.1T–1.0T].
Fig. 6 shows that the percentage of failures grows with t_A in all the targets. In buses and combinational logic, the increase is roughly logarithmic (note that the scale of t_A is logarithmic), with higher values of $P_{Failures}$ for buses.
In the storage elements, a much smoother slope is observed. This is caused by two facts: i) faults in critical registers provoke failures independently of t_A, and ii) faults in memory mainly cause latent errors, because the faults are injected randomly and the memory is much bigger than the workload.
Globally, intermittent faults in buses are the most damaging, except for low values of t_A, where faults in the storage elements provoke more failures.
Fig. 6 also shows that multiple faults have much more impact than single faults. This is a predictable behavior, as multiple faults affect simultaneously various physical locations of the system. Values of $P_{Failures}$ over 90% can be seen for intermittent multiple faults in buses.
Fig. 7 reflects that P_{Latent} does not show a uniform trend with regard to t_A. It rises slightly in combinational logic and buses, and it is almost constant in storage elements. This discrepancy is due to the fact that faults in memory and registers propagate “instantaneously”, while faults in combinational logic and buses can be masked (by logic or temporal masking mechanisms). In this case, the rise of t_A reduces the effectiveness of fault masking.
We can also observe in Fig. 7 that latent errors are much more frequent in storage elements than in the other targets. This is due to both the absence of masking effects in the propagation, and the existence of a great quantity of cells, especially in memory, that can be perturbed but are not accessed later by the workload.
P_{Latent} shows little differences between single and multiple faults in buses and combinational logic. In the storage elements, single faults provoke even more latent errors than multiple faults, because most of multiple faults lead to failures.

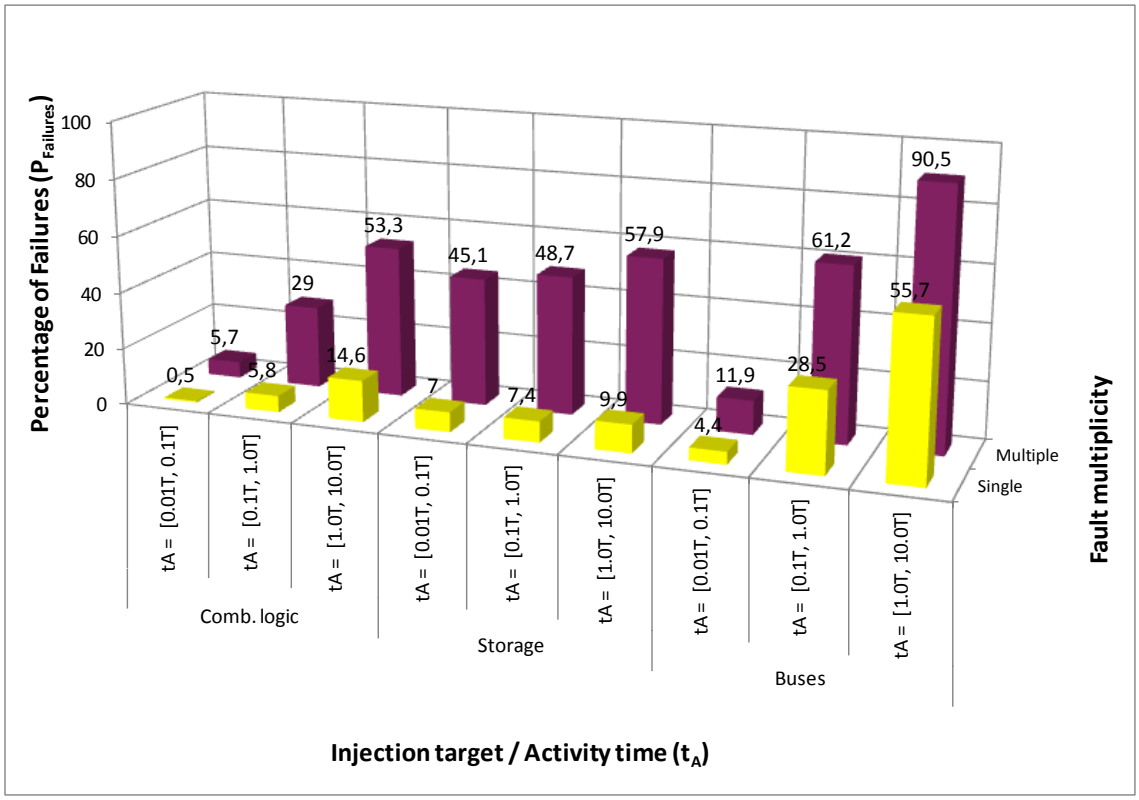


Fig. 6. Influence of the activity time and the fault multiplicity in the percentage of failures.

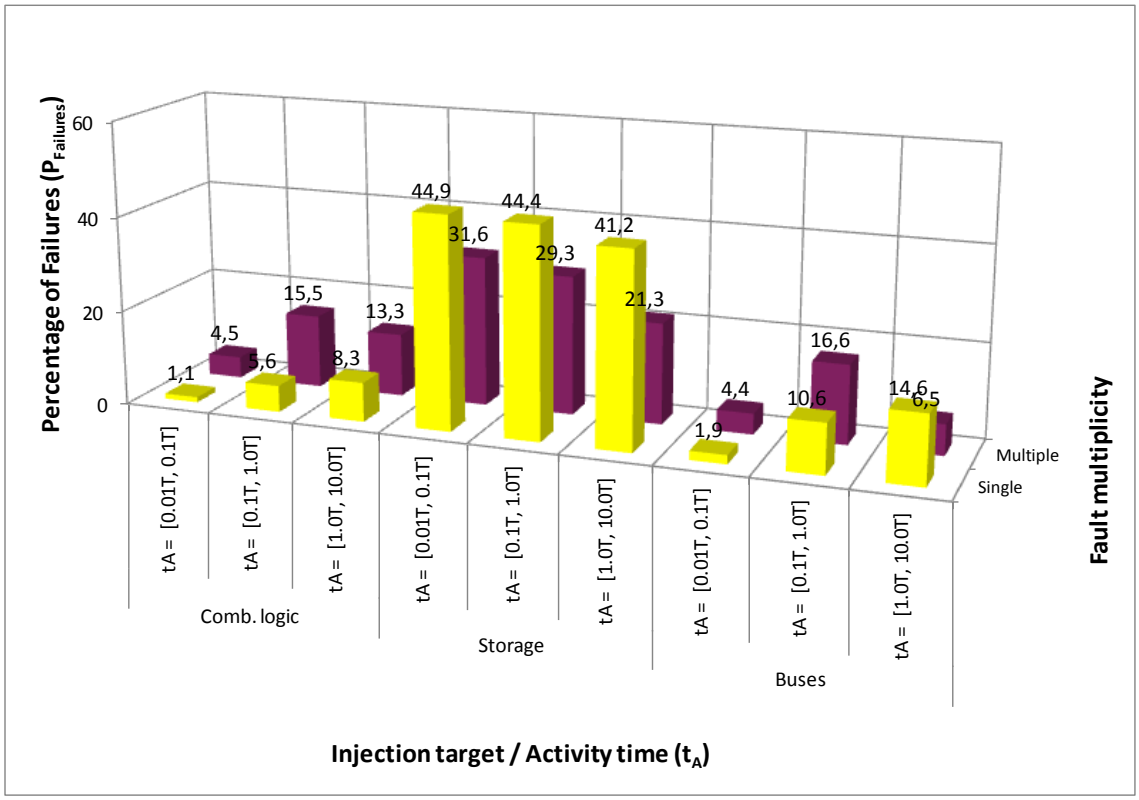


Fig. 7. Influence of the activity time and the fault multiplicity in the percentage of latent errors.

Inactivity time

Unexpectedly, no significant influence has been observed when varying t_I. A more detailed analysis has shown that the separation between activations has not changed the total number of activations in the bursts because the workload is long enough to fit all activations. Thus, the system is affected regardless of t_I.

Burst length

In previous experiments, L_{Burst} was varied randomly in the range [1–10]. Fig. 8 shows the results obtained when fixing this parameter from 1 to 10, with t_A and t_I defined in the intermediate range [0.1T–1.0T]. The figure shows the results for *multiple* faults.

In the buses, $P_{Failures}$ rises asymptotically up to 75%, stabilizing from $L_{Burst} \approx 9$. In combinational logic, the trend is also asymptotic, rising up to 34-35% from $L_{Burst} \approx 7$.

In the storage elements, $P_{Failures}$ presents a nearly constant behavior, with small variations between 48.5% and 54%. This is due to:

- Faults affecting critical registers provoke a failure in the firsts activations (i.e., for low values of L_{Burst}).
- Faults affecting unaccessed memory cells only cause latent errors, but not failures, even in the presence of several activations.

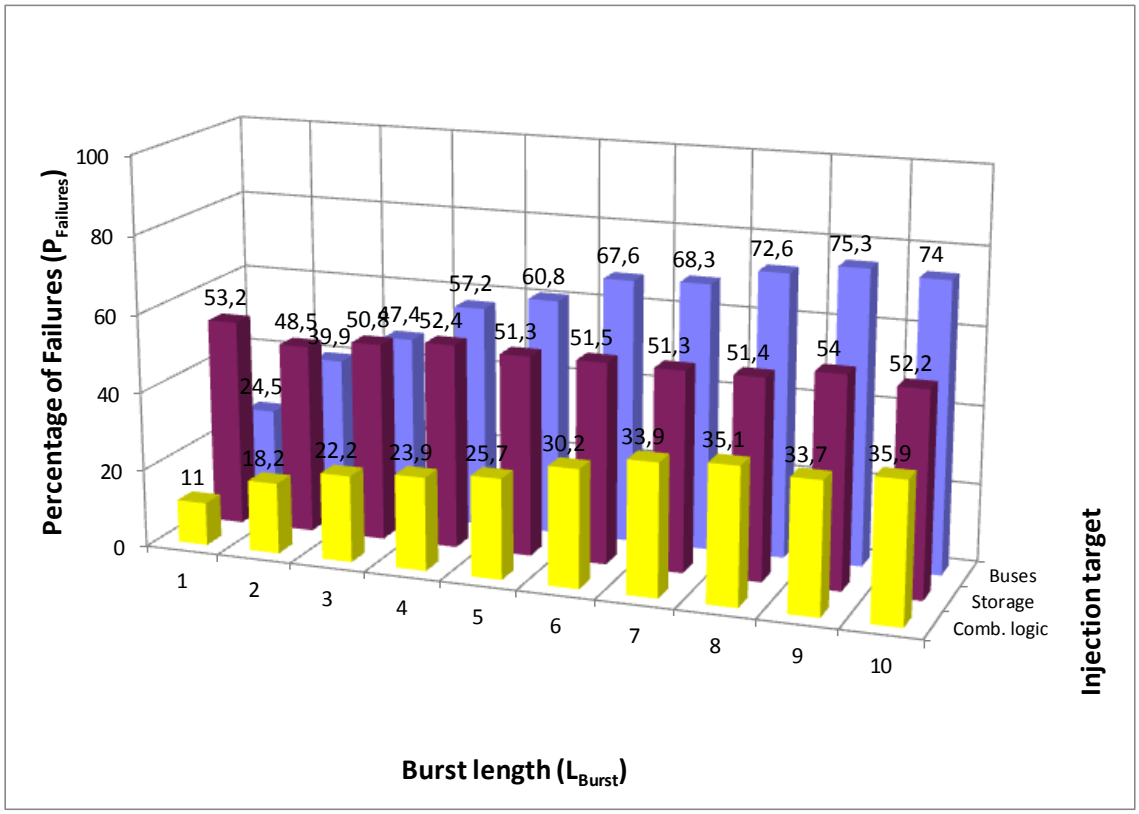


Fig. 8. Influence of the burst length in the percentage of failures (multiple faults).

4.2. Influence of the injection target

Previous section has shown that:

- Intermittent faults in buses are really harmful, as buses are a bottleneck in the fetch and execution phases of the microcontroller.
- Intermittent faults in registers provoke a high percentage of failures, because they store intermediate results when executing an instruction.
- Faults in memory manifest mainly as latent errors.
- Combinational logic is less sensitive, although its impact can be notable for high values of t_A and L_{burst}.

4.3. Influence of the system clock frequency

Fig. 9 shows how the frequency influences in the percentage of failures, for single and multiple intermittent faults in buses. Frequency has been varied from 1F to 10F, where the nominal frequency is F = 10 MHz. It can be observed that $P_{Failures}$ rises asymptotically, similarly to as with the burst length. When injecting single faults, $P_{Failures}$ tends to 60%, while with multiple faults it seems to stabilize over 90%. The asymptote is attained roughly at 7F (i.e., about 70 MHz).

The same growing asymptotic trend has been observed in storage and combinational logic targets, but with lower values of $P_{Failures}$.

The explanation of the notable frequency influence lies in the fact that at higher frequencies, the probability of capturing an error in active edges of synchronous components (like memory and registers) rises. It is important to remark that the frequency increase has been a trend in VLSI technologies.

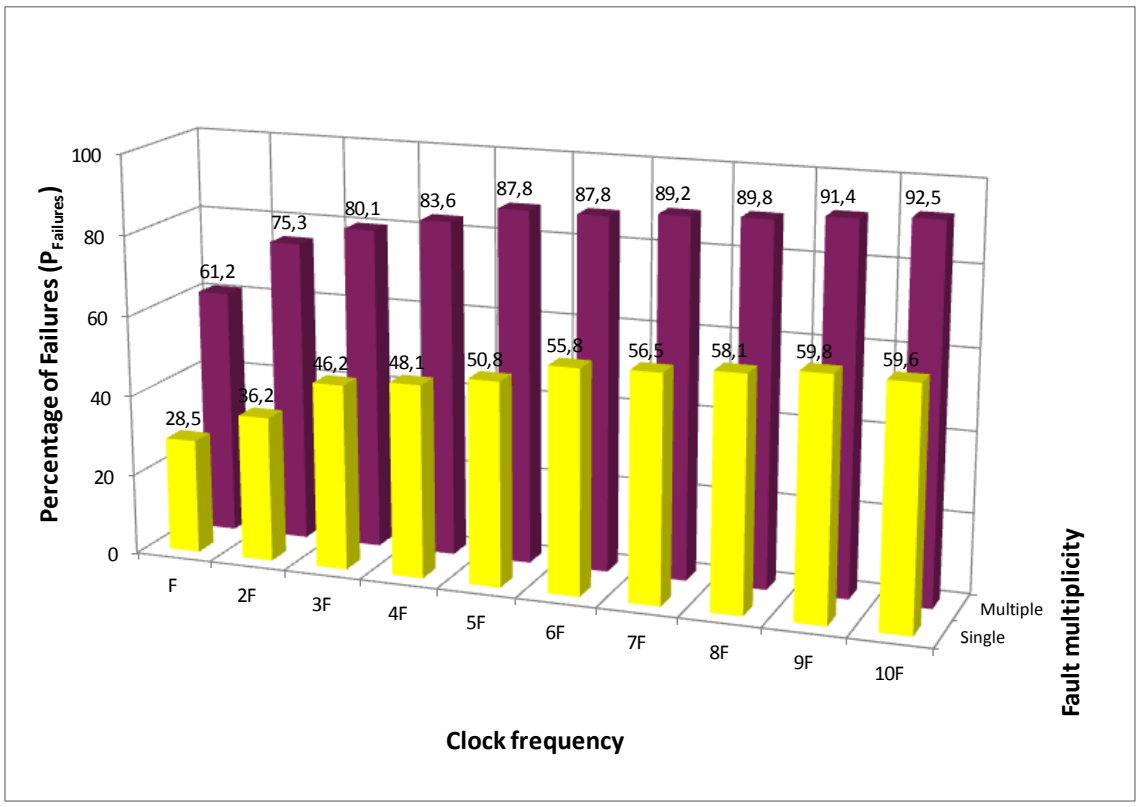


Fig. 9. Influence of the clock frequency in $P_{Failures}$ (Target: buses).

4.4. Comparison to transient and permanent faults

In [17] we compared the effects of intermittent, transient and permanent faults in the storage elements and buses. Fig. 10 completes this study showing the impact in the combinational logic. Transient fault models injected were *pulses* (to emulate Single Event Transients, or SETs), and *indetermination* (undefined logic value provoked by voltage and current variations) [2]. For permanent faults, the fault models injected were *stuck-at(0,1)*, *open* and *indetermination* [2, 3]. In the case of transient faults, three different ranges of fault duration have been considered. In fact, the same used for the activity time in intermittent faults. From the figure, we can see that intermittent faults provoke a greater percentage of failures than transient faults. This is an expected result, as a burst of intermittent faults manifests like a sequence of transient faults in spite of having different origin. The greatest impact corresponds to permanent faults because of their infinite duration, although this value is a bit bigger than that provoked by intermittent faults with the greatest t_A range. The same trend was found in [17] for storage elements and buses.

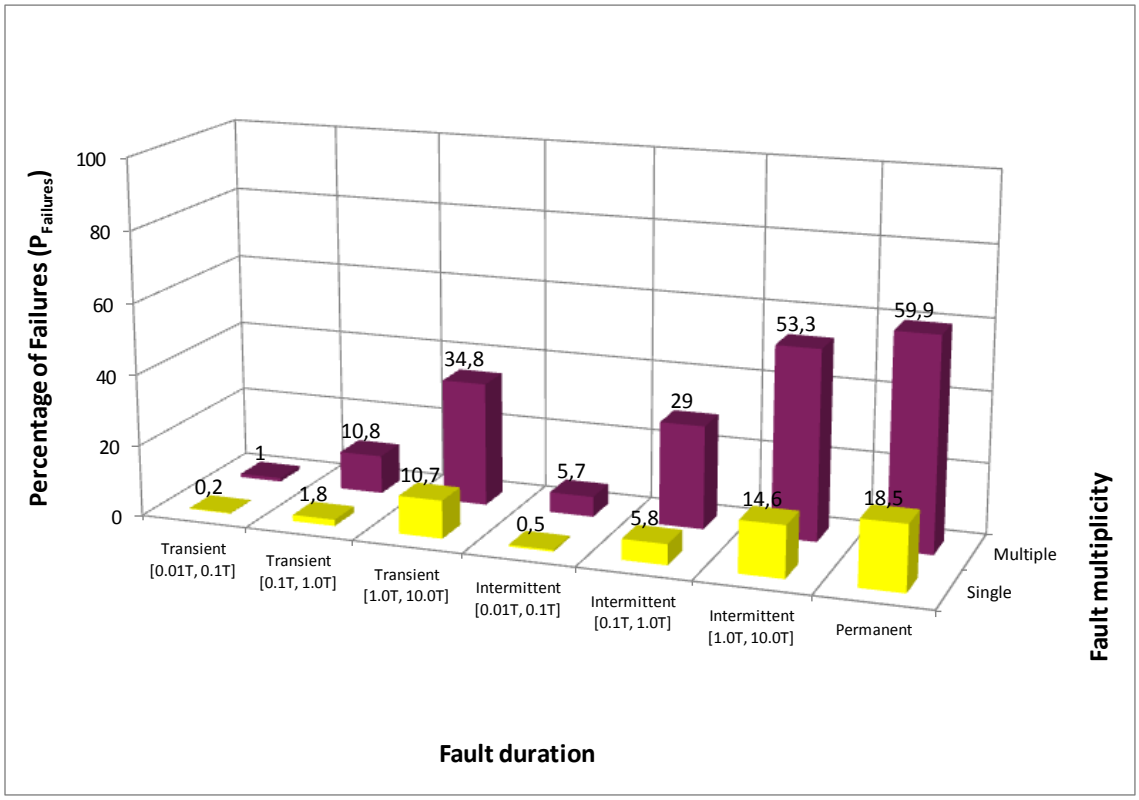


Fig. 10. Comparison of the effects of transient, intermittent and permanent faults (target: combinational logic).

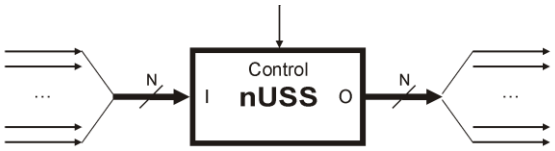
4.5. Extending the intermittent fault models by means of *saboteurs*

As commented before, the VHDL model of the microcontroller does not present any information about delays. This is common in core models, as delays are introduced in the implementation phase, after *place and route*. As reflected in Table 1, *intermittent delay* fault model is quite important. So, in order to inject this type of fault, *saboteurs* technique has been used instead of *simulator commands*. This technique can apply more fault models but at the cost of modifying the VHDL model.

A *saboteur* is a special VHDL component added to the original model [3, 18]. When activated, the mission of this component is to alter the value or timing characteristics of one or more signals, simulating the occurrence of a fault. During the normal operation of the system, instead, the component remains inactive.

The study has been focused on the buses, which are critical locations where intermittent faults have been observed (see Table 1). *Saboteurs* have been placed in the ports of the components connected to the buses. The model of the *saboteur* applied is *n*-bit uni-directional (nUSS) [20]. Generally, *saboteurs* can be applied in structural architectures, made of interconnected components.

As an example, Fig. 11(a) shows the scheme and a pseudo-code implementation of the nUSS. *I* and *O* are respectively the input and the sabotaged output of the *saboteur*. *Control* signal determines the instant when the *saboteur* has to be activated, and the duration of the activation. Depending on the number of activations and deactivations of this signal, permanent, transient or intermittent faults can be injected. The function F_{inj} implements the injection of the fault in the targeted signals. The operation performed depends on the fault model to be injected, as shown in Fig. 11(b). The fault is injected in the rising edge of the control signal, and inputs are ignored during the fault duration. If the control signal is activated repeatedly during t_A, and deactivated during t_I, intermittent faults can be injected.




```
architecture behaviour of nUSS
signal inj_cfg;

begin
  process (I, Control)
  begin
    if Control = '1' and not Control'stable then
      inj_cfg <= injection_configuration();
      O <= F_inj(I, inj_cfg);
    elsif Control = '0' then
      O <= I;
    end if;
  end process;
end architecture;
```

(a)

Fault model	F_{inj}(I, inj_cfg)
Stuck-at 0	'0'
Stuck-at 1	'1'
Bit-flip, Pulse	not(I)
Open	'Z' (high impedance)
Delay	I after delay, delay > 0
Indetermination	'X'
Short	I(i) = I(j)
Stuck-open	'0' after $t_{retention}$

(b)

Fig. 11. nUSS *saboteur*. (a) Scheme and implementation. (b) Injection function.

Fig. 12 illustrates how a generic internal component of a microcomputer is sabotaged. *Saboteurs* are inserted between the buses (address, data, and control) and the component to be perturbed.

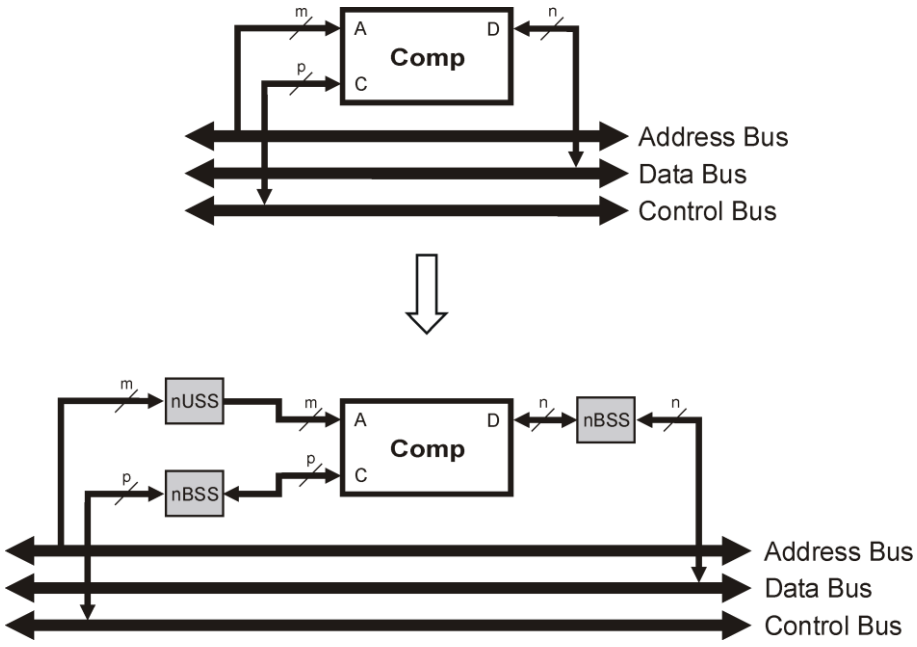


Fig. 12. Example of sabotaging an internal component of a microcomputer.

The main injection parameters of the experiments carried out in this section are:

a) Injection targets: Microcontroller (8051) buses. The following components have been sabotaged: ALU, Control Unit, ROM and internal RAM.
b) Injection technique: *Saboteurs.*
c) Fault models: *Intermittent* {*Pulse*, *Short*, *Open*, *Delay*}. Notice that, concerning the previous experiments, the fault model set has been extended with *Intermittent Short* and *Intermittent Delay*.
d) Values of delay (only for delay fault model): Defined according to a Uniform distribution function in the range [0.1T, 1.5T], being T the nominal system clock cycle (100 ns).
e) Workloads: Arithmetic series, Bubblesort algorithm and Matrix multiplication. Notice that also workloads have been extended.
f) Burst parameters:
   - t_A and t_I are defined according to a Uniform distribution function in the intermediate range [0.1T–1.0T].
   - L_{Burst} is defined according to a Uniform distribution function in the range [1–10].

Fig. 13 compares the impact (in terms of percentage of failures) of the new fault load. We can observe that the values corresponding to the *intermittent pulse* are bigger than those obtained with *simulator commands* technique shown in Fig.6. The reason is that *saboteurs* have injected a higher percentage of faults in the data bus, an especially critical target. When inserting the *saboteurs* in the model, the number of injection targets that affect the data bus has been increased.

From the figure, some conclusions can be extracted. *Intermittent pulse*, *Intermittent open* and *Intermittent short* fault models present similar impact levels, except in some cases where *Intermittent pulses* are a bit more damaging. This suggests that *Intermittent opens* and *Intermittent shorts* manifest mostly as *Intermittent pulses*. On the other hand, notice that *Intermittent pulse* fault model is easier to implement, as it can be injected with *simulator commands* technique, whereas *Intermittent open* and *Intermittent short* should be implemented by using *saboteurs*.

With regard to *Intermittent delay* fault model, its impact is quite lower. This fact is caused by timing masking phenomena in synchronous components. The implementation of this fault model requires the use of the *saboteurs* technique, implying in this way an extra overhead. However, it is expected an increasing incidence of timing errors in deep submicron technologies due to the reduction of the metal layer widths and the rise of clock frequencies, so this fault model should be taken into account. More details about fault model extension using saboteurs can be found in [21].

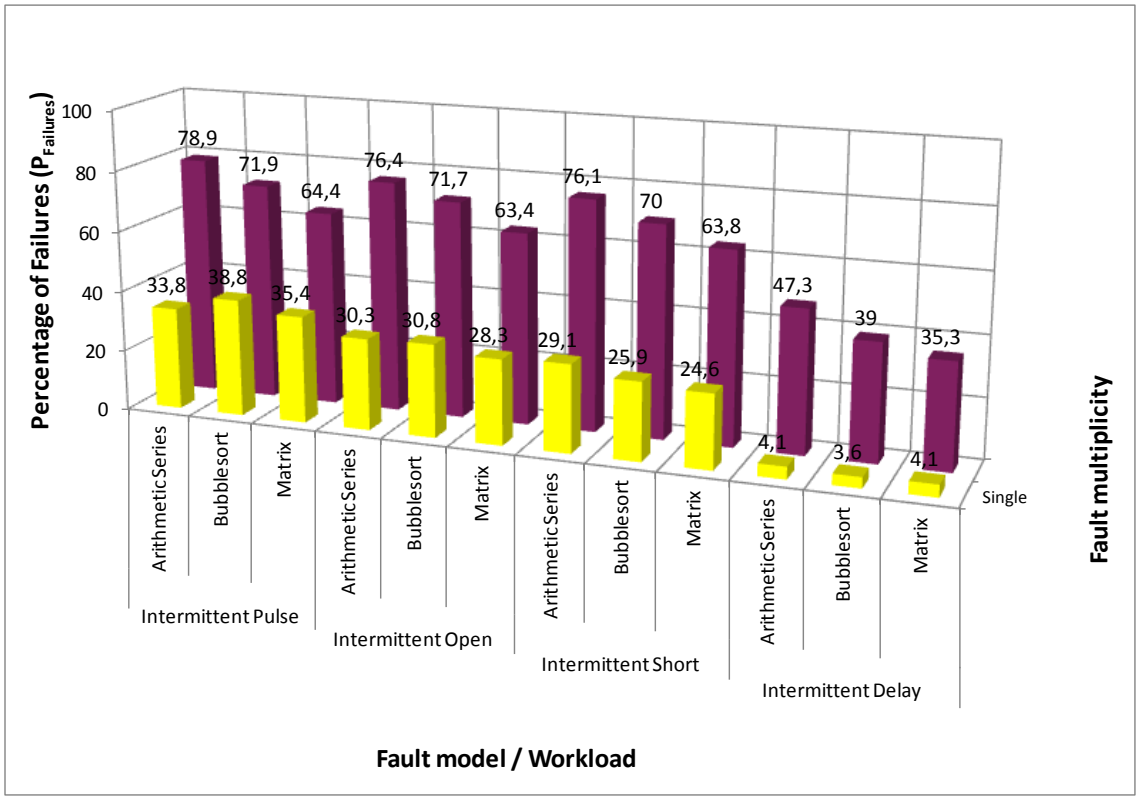


Fig. 13. Percentage of failures provoked by extended fault load using *saboteurs* (target: buses).

5. Conclusions

In this work we have presented a case study of the effects of intermittent faults on the behavior of a commercial microcontroller. The methodology used lies in VHDL-based fault injection technique, which allows a systematic and exhaustive analysis of the influence of different fault parameters.

In general terms, and assuming the important influence of the microcontroller and the workload used, some conclusions can be extracted.

Related to the influence of burst parameters:

- The *activity time* is a quite important parameter. Increasing the duration of the activations provokes a significant rise of the percentage of failures, especially in buses and combinational logic. Its influence in latent errors is lower. This is the common behavior of some intermittent faults (for instance, those due to wearout), as they evolve to permanent.
- The *burst length* has also a notable influence. When increasing, the percentage of failures grows asymptotically. Like the *activity time*, the *burst length* of intermittent faults that become permanent tends to grow.
- The *inactivity time* has not shown any significant impact because the duration of the workload was long enough to fit all activations. It is expected that this parameter will be more important in a fault-tolerant system, as the separation between activations may affect the detection and recovery latencies.

Regarding other factors:

- The spatial multiplicity of intermittent faults presents a significant impact on the behavior of the system. Multiple faults provoke a much greater percentage of failures than single faults. The presence of multiple intermittent faults is an expected trend in deep submicron technologies, as the feature size of the manufacturing process reduces.
- The injection target leads to important differences. Buses and registers are the most sensitive targets to intermittent faults, while faults in memory provoke mainly latent errors. On the other hand, the impact of faults in combinational logic is also important. As the effect of masking mechanisms reduces in deep submicron technologies, it is expected that combinational logic will be increasingly sensitive to intermittent faults.

- The rise of the clock frequency increases the impact of intermittent faults. This is due to the higher probability of capturing errors in the active edges of synchronous components. The percentage of failures presents an asymptotical rising trend, similar to the burst length dependency. The increase of the clock frequency has been a trend in deep submicron technologies.

Also, the impact of intermittent faults has been compared to that of transient and permanent faults. Intermittent faults provoke a quite greater percentage of failures than transient faults. The greatest impact is caused by permanent faults because of their infinite duration, although intermittent faults with long activations present a similar impact.
Finally, the fault model set has been extended using the *saboteurs* technique. In this way, intermittent *delay* and *short* faults could be injected. It is important to include *Intermittent delay* in the fault model set, because it is expected an increasing incidence of timing errors in deep submicron technologies.
The results obtained in this work suggest the suitability of adding mitigation techniques to deliver fast error detection and correction of intermittent faults in buses and critical registers, such as hardware-implemented ECCs. It is expected also that mitigation techniques in combinational logic will be increasingly necessary.
In the future, we have planned to extend this study in different aspects. Firstly, we want to analyze the effect of intermittent faults on more complex microprocessors and fault-tolerant systems. Secondly, going more deeply into the causes and mechanisms of intermittent faults in new submicron technologies, in order to propose new fault models related to them. In addition, it would be interesting to use different probability distributions to model more accurately wearout mechanisms. Finally, we intend to study mitigation techniques for fast detection and recovery of intermittent faults.

6. Acknowledgement

This work has been funded by the Spanish Government under the research project TIN2009-13825.

7. References

[1] C. Constantinescu, “Trends and Challenges in VLSI Circuit Reliability”, IEEE Micro, 23(4):14-19, 2003.
[2] P.J. Gil et al., “Fault Representativeness”, Deliverable ETIE2 of Dependability Benchmarking Project, IST-2000-25245, 2002.
[3] A. Benso and P. Prinetto, eds., “Fault Injection Techniques and Tools for VLSI reliability evaluation”, Kluwer Academic Publishers, 2003.
[4] C. Constantinescu, “Impact of Intermittent Faults on Nanocomputing Devices”, in Procs. DSN 2007 Workshop on Dependable and Secure Nanocomputing, UK, 2007, pp. 238–241.
[5] M. Stanisavljevic, M. Schmid and Y. Leblebici, “Reliability of Nanoscale Circuits and Systems. Methodologies and Circuit Architectures”, Ed. Springer, 1st Edition, 2011.
[6] J. Arlat et al., “Fault Injection for Dependability Validation: A Methodology and Some Applications”, IEEE Transactions on Software Engineering, 16(2):166–182, February 1990.
[7] "IEEE Standard VHDL Language Reference Manual", IEEE Std 1076-1993.
[8] D.P. Siewiorek and R. S. Schwarz, “Reliable computer systems: Design and evaluation” Ed. Digital Press, 3rd Edition, 1998.
[9] C. Constantinescu, “Dependability Benchmarking using Environmental Test Tools”, in Procs. Reliability and Maintainability Symposium, Alexandria, VA, USA, 2005, pp. 567–571.
[10] J.H. Stathis, “Physical and Predictive Models of Ultra Thin Oxide Reliability in CMOS Devices and Circuits”, in Procs. 39th Annual IEEE International Reliability Physics Symposium, Orlando, FL, USA, 2001, pp. 132-150.
[11] S. Borkar et al., “Parameter Variations and Impact on Circuits and Microarchitecture”, in Procs. 40th Annual Design Automation Conference, Anaheim, CA, USA, 2003, pp. 338 342.
[12] J.W. McPherson, “Reliability challenges for 45nm and beyond”, in Procs. 43th Annual Design Automation Conference, San Francisco, CA, USA, 2006, pp. 176–181.
[13] J.C. Smolens et al., “Detecting Emerging Wearout Faults”, in Procs. of the IEEE Workshop on Silicon Errors in Logic - System Effects, Austin, TX, USA, 2007, pp. 276–287.
[14] S. P. Park and K. Roy, “Reliability Implications of Bias-Temperature Instability in Digital ICs”, IEEE Design & Test of Computers, 26(6): 8–17, 2009.

[15] M. Bashir and L. Milor, “Modeling Low-k Dielectric Breakdown to Determine Lifetime Requirements”, IEEE Design & Test of Computers, 26(6):18–26, 2009.
[16] J. Gracia et al., “Analysis of the influence of intermittent faults in a microcontroller”, in Procs. 14th IEEE Design and Diagnostics of Electronic Circuits and Systems, Bratislava, Slovakia, 2008, pp. 80–85.
[17] D. Gil et al. “Injecting Intermittent Faults for the Dependability Validation of Commercial Microcontrollers”, in Procs. IEEE International High Level Design Validation and Test Workshop, Nevada, USA, 2008, pp. 177–184.
[18] E. Jenn et al., “Fault Injection into VHDL Models: The MEFISTO Tool”, in Procs. 24th International Symposium on Fault-Tolerant Computing, Austin, TX, USA, 1994, pp. 66–75.
[19] http://www.oregano.at
[20] J.C. Baraza et al., “Enhancement of Fault Injection Techniques Based on the Modification of VHDL Code”, IEEE Transactions on VLSI Systems, 16(6): 693–706, 2008.
[21] J. Gracia-Morán et al., “Searching representative and low cost fault models for intermittent faults in microcontrollers: a case study”, in Procs. IEEE Pacific Rim on Dependable Computing, Tokyo, Japan, 2010, pp. 11–18.

Highlights

- To generate intermittent fault models at logic and register transfer (RT) abstraction levels, paying special attention to faults in combinational logic.
- To inject these fault models, applying different VHDL-based fault injection techniques.
- An exhaustive variation of the fault and system parameters, in order to obtain an extensive analysis of their effects.
- A study of the impact of intermittent faults in the behavior of a commercial microcontroller.
- Comparing the influence of intermittent faults to that of transient and permanent faults.