# Reliability in modern processors

## An overview into dealing with a life of soft faults in modern VLSI.

*Ricardo Constantino Mendes*

*Mestrado em Engenharia de Sistemas Computacionais Críticos, ISEP,*
*30 December 2022*

## Abstract

Wear-out and faults are inevitable during the life of any system. The word *mitigation* is recurrently eared when describing techniques to reduce the impact of these faults, a clear outcome of the second law of thermodynamics. Being a law of nature, *mitigating* and *tolerating* its consequences is the only realistic goal.

This paper aims to introduce the theme of *reliability in modern processors* by surveying the recent papers, articles and conferences written about this field of study. The paper wants to present a wide range of techniques that tolerate faults and errors in a system.

Given its academic and introductory nature, it does not present sharp limits between the proposed sub-topics for the class assignment: wear out, hard faults and soft faults.

Nonetheless, the paper's subtitle tries to steer the discussion toward the last sub-topic.

During the investigations, a slight shift in perspective or the necessity to introduce an analogy to clarify some concept adds to the text loose ends but a broader view of the problem that we pretend to explore here.

**Keywords:** VLSI, fault-tolerance, hard faults, soft faults, reliability.

## Introduction

The *reliability* of all the components that structure our information-driven society is a crucial discussion that has gained continuous relevance in the academic and research community.

Suppose we add extreme computational and environmental requirements, such as *real-time systems* and *space exploration*. In these cases, we will find a vital field of research with a strong impact at the forefront of exciting industries like avionics, self-driving cars and space exploration.

For this reason, we propose to analyze the most common strategies to deal with faults and errors and describe their main characteristics.

In the last few decades, the *miniaturization*, increase in density, and mass production of Very Large Scale Integration (VLSI) circuits, have exacerbated the reliability problem in complex systems.

Different levels of abstraction are used to better understand and control this complexity, like boolean logic, high-level software patterns, or even fundamental laws of physics and electromagnetism.

As we will see, some strategies are particular to one of these layers. However, we do not categorize them through these criteria.

Also, some faults are particularly more acute in some components like flash memories than in SRAM's[1], for example. The same applies to other circuit parts, e.g., *Arithmetic Logic Units* (ALU) or COMS. Unlike RAM, which can be protected by error-correction codes (ECCs), elements like latches and flip-flops are more challenging[2]. However, we will not focus on these distinctions.

The fault-tolerant strategies that will be described can be component-specific. However, most have broader principles that can be understood without diving into the most granular details. That is why we aim to a more ample exposure to the state of the art.

### Paper structure

The paper is structured into four chapters.

The first chapter clarifies the main concepts and terms used when addressing reliability in modern processors. In addition, some other terminologies – not directly related to the paper – are being discussed because of their relevance and helpfulness in introducing the subject.

The second chapter focus on the problem that puts into question the *reliability* of integrated circuits (IC) in general. We also try to answer the *why* its root causes.

The third chapter describes the state of the art of different mitigation and fault-tolerance techniques published in the last ten or fifteen years.

Finally, the fourth chapter concludes the paper by giving an overview of the research results and trying to expose and identify some of the main trends in the field.

# 1. Main concepts

## 1.1 Faults, errors and failures

In the context of reliability, *faults* can be subdivided into two groups: *hard faults* and *soft faults*. The first derives from defects in the circuit and the last from errors caused by environmental conditions.

The occurrence of a fault has its consequences. Usually, this cause and effect is explained in a chain of threats that will be useful to expose here for further exploration of our paper's topic.

A bit-flip caused by a *single-event-upset* (SEU) that alters some output of an ALU can be identified as a fault. This fault can result in an error if it changes the correctness of the output. If the error is not identified and rectified and is propagated to other parts of the system, we have a failure. Here, the rest of the system can be put into jeopardy.

… fault → error → failure …

## 1.2 Hard faults

Hard faults – also known as permanent faults – refer to physical anomalies in the device or circuit. They are usually irreversible and cause long-term malfunction[3]. A sub-group of hard faults, called intermittent faults, are sometimes mentioned in this field of research and are described as occurring in bursts when the proper environmental conditions arrive.

Figure 1 shows the daily number of single-bit errors observed in one system. The burst of error seen in the graph possibly shows the existence of an intermittent fault[8]. The other more random data point is probably the result of soft faults.

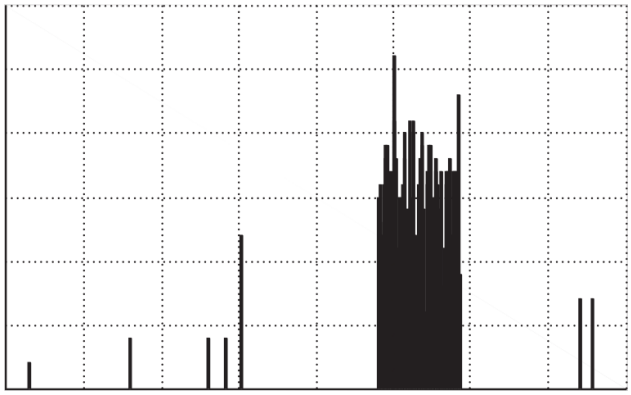

*Figure 1 -Daily number of memory single-bit errors reported by one system over 16 month.*

Being both hard faults, permanent- and intermittent faults derive from physical defects in a circuit. However, in the first one, this leads to a dramatic failure making the system almost always unusable. In the second one, the consequences of the defect only arrive in certain conditions but consistently and repeatedly in the exact location[4].

There are many causes for hard faults, such as:

> *Electro-migration, which causes thinning and eventual open circuits of metal tracks. Hot carrier effect, which causes shift in device threshold voltage and it does convey conductance. Time dependent dielectric breakdown, which causes gate oxide to substrate short circuit.*[3]

The majority of the semiconductor industry already adopts solutions like copper interconnects instead of aluminum ones, mitigating the negative impact of hard faults derived from electro-migration – as copper provides a higher threshold of current-induced atomic transport generated by collisions of electrons with metal atoms[4].

Although the softening in the growth rate of the density of transistors in an integrated circuit – maybe the end of Moore's Law –, that led to a shrinking of geometries, lower power voltages and higher frequencies, the semiconductor industry has improved at a superior rate, through innovations in design and manufacturing techniques, the overall yield[5].

Models of prediction of the yield of a semiconductor device – *the ratio of the number of products that can be sold to the number of products that can be manufactured*[6] – also have a significant impact, not only due to economic reasons but also to quantify the slow-dow and standards in processors architectures. A defect-free product is impossible, but a lower margin of mainly intermittent faults can quickly be assured.

Some simulation techniques – *like statistical fault injection, analytical models, and performance models*[7] – that have been adopted and improve the outcomes during a product release are a field of study in itself and a constant presence in the production of IC.

## 1.3 Soft faults

An interesting interconnection that can be made between intermittent- and soft-faults states as follow: the replacement of the defective circuit removes the intermittent faults; this does not happen to soft faults[4].

This beautifully shows the nature of this type of fault: they are caused by a temporary environmental condition ranging from temperature to voltage, from cosmic rays – high-energy particles – to electromagnetic interference. That is why they are also known as *transient faults.*

The current use of lower supply voltages and higher VLSI increases the rate of occurrence of this type of error[8].

Single-bit faults are the most common soft faults deriving from high-energy particles. This does not mean that more extensive damage cannot happen. Hard faults like *gate ruptures* were already identified as resulting from cosmic radiation.

There are multiple outcomes from soft faults that can help us better understand some strategies to improve fault-tolerance.

The most dramatic outcome would be a *silent data corruption* (SDC), where an error is not detected and also not corrected. Another outcome states that an error is detected but not corrected. This is usually categorized as *detected unrecoverable errors* (DUE)[9].

We should remember the interchangeable relation between soft error *avoidance*, *detection*, and *recovery* when discussing fault-tolerant techniques.

An interesting example is the action of restoring a system from an error by reprocessing some instructions. This comes at a cost: reprocessing takes time and energy, resources that usually have fragile margins.

Depending on the given requirements, selecting the proper fault-tolerant technique must consider this type of consequence and trade-offs.

As we will see, a large toolbox is available when dealing with this issue.

## 2.3 Wear-out and useful lifetime

As we know and will reinforce during the paper, faults and errors are inevitable.

Coming from external environmental conditions or internal system dispositions, the rate of error a system is prone to, dictates its reliability, purpose, and, in more generalized terms, its usefulness.

All systems tend to disorder, biological or mechanical, and the wear-out of a VLSI reflex this.

Some manifestations, like phenomenons named *Bias Temperature Instability, Hot Carrier Injection, and Time Dependent Dielectric Breakdown,* have a direct negative impact on circuit reliability and are repeatedly mentioned in specialized bibliography [10].

The prediction of the lifetime – or, more specifically, the useful life – of a system is of great importance to reliably map a the system's requirement to a specific architecture, and fault-tolerance technique.

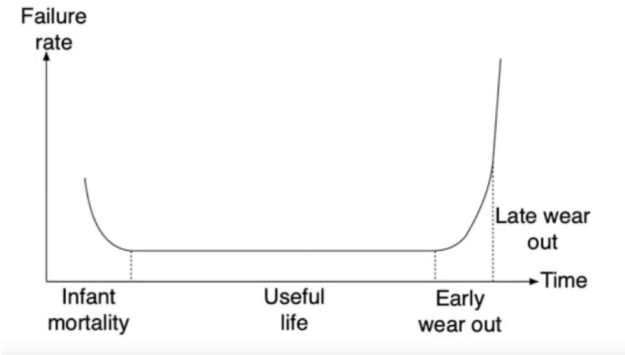


*Figure 2 – Integrated circuit rate of failures during its lifetime.*

The wear-out starts when a system is released in the market, see Figure 2.

During the first phase, named here as *infant mortality*, most of the systems, in this case, a generic IC, will display some defects from manufacturing or design decisions. Most of these failures are intermittent faults since the most permanent hard faults are usually identified and dealt with in a pre-release phase.

In the next phase of *useful life*, it is noticeable that the failure rate is not zero but low and constant. Most of these failures have their origin in soft faults.

In the *wear-out* phase, we observe an increase in failure rate mainly from hard faults. This increases until the IC is unviable.

For a processor in the market, the implementation of hard-fault-tolerance aims to extend the system's lifetime and yield during manufacturing.

However, in this paper, we will focus on the *useful life* of a VLSI and the techniques to mitigate errors and failures that occur from soft faults.

# 2. The problem of soft faults

## 2.1 Context

Intensive scaling for VLSI circuits is a crucial strategy to improve computation performance. Unfortunately, this also means an increase in aging degradation[10] and the likelihood of soft errors.

The miniaturization of circuit components to submicron levels increases the sensitivity to charge collection due to particle strikes[11] and the increase in transistor density means a greater probability that a single strike affects multiple components – a s*ingle-event-multiple-upset* (SEMU).

> *Measurements showed that the effect of alpha particles increased the soft error rate 30 times as the manufacturing process went from 0.25 micron to 0.18 micron and supply voltage dropped from 2 V to 1.6 V. At the same time, the neutron's impact increased the soft error rate by 20 percent*[12].

If we contextualize this to the exponential increase in the number of devices – see Figure 3 – that drives the IoT revolution, we can see the relevance of this research field.

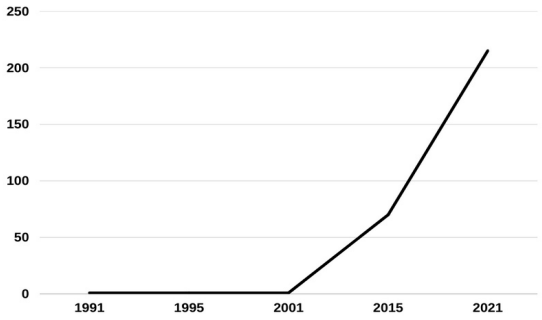


*Figure 3 – Number of devices with ARM processors (in billions). Source: Talk given by Alexandre Peixoto Ferreira, researcher in ARM RSH, October 2022, ISEP.*

The problem isn't new, being already identified and studied during the the 60's and 70's[13].

IoT and self-driving vehicles, in connection with machine learning and AI , are thinning the line that separates the virtual world from the real world.

The responsibility inherent in the collision of both worlds gives rise to critical and life-threatening situations. The systems must be predictable and their outcome deterministic. These requirements found in *real-time systems* are impacting our everyday lives.

The tendencies that show a slow elimination of the human level as a fallback mechanism require the implementation of fault-tolerant techniques[14]. The holly grail of a level 5, fully autonomous driving vehicle is the prime example.

Before the emergence of the motorcar, the horse carriage added a fault mechanism implicitly, e.g., if the driver fell asleep, the horse would, in normal circumstances, never throw itself into a tree. With the motorcar, the driver needed to be on constant alert[15]. Nowadays, we are trying to return to the old days, but instead of a horse, we are looking for reliability in our systems.

Another particular case is space exploration. The tendency to send robots instead of people to space is noticeable in the great date when the last human touched the moon's surface, on the 11th of December 1972.

Besides the clear objective of reducing the risk of losing a human life, robots are more versatile in outer-space to gather data and withstand long missions. As a curious species, space exploration is increasing and consequentially pushing the limits of onboard electronics.

The trend shows an apparent increase in the amount of data processed during a mission. Additionally, most on-board electronics need to be reprogrammable for mission uncertainties. The requirements are rigorous, floating between maximizing the processing capacity and the level of reliability of the overall system, something that, if overlooked, could abort the entire mission[16].

If we want to be more succinct, all high-energy and charged particles are the primary source of faults and errors. Most of the faults in this type of environment originate in radiation, in particular *Cosmic Rays*, *Solar Winds* and *Van Allen's Belt*. These errors are known as *Single Event Effects* (SEE). Despite mostly giving rise to soft errors, hard errors can also occur in events called – and self-explanatory – *Single-Event Latch-up* and *Single-Event Gate Rupture*.

# 3. Fault-tolerance techniques

As we will see, fault *avoidance* relies on *improved materials, manufacturing processes, and circuit design* [17]. In return, fault-*tolerance* is implementable at the circuit or architecture level. It is based on the capacity to *self-diagnose*, *evaluate* its availability – reliability – and be able to *correct* or *recover* from a errors[14].

These tree capabilities will be present in all the described techniques.

Fault-tolerance techniques can be classified into several categories. The primary branching will separate *architectural solutions* from *circuit solutions.* In some bibliography[18] a third solution can be found that highlights the manufacturing techniques – *process solutions*.

The term *redundancy* will be a constant in most of the techniques. They can be *spacial* or *time-based*, but nothing prevents some kind of hybrid solution.

Also, the techniques will sometimes be described in a specific context, i.e., given the requirements of a system, usually a solution can be more suitable in comparison with another that would, on paper, look more suitable.

This will also be stated in the following chapters.

## 3.2 Circuit Techniques

### 3.2.1 Radiation Hardness-By-Design (RHBD)

One of the more obvious techniques is reducing the circuit's sensibility to radiation.

As already mention, the lower voltage of modern processors just increases the rate of soft-faults by reducing the energy necessary to change a state for a given transistor. The current lower voltage seen in most processors derives from the necessity to reduce power

consumption so that heat dissipation during computation is not a problem.

Also faster circuits correspond to smaller margins between cycles and smaller time-frames that could smoothen the impact of high energy particles.

RHBD and RC-based techniques (circuit containing resistance and capacitance) impose most of the cases a solution that comes with a big performance penalty.

An example found for SRAM cells is to:

> … *substantially increase the storage node capacitance in each cell without any area impact by introducing a Metal-Insulator-Metal (MIM) capacitor between the polysilicon and Metal 1 layers of the cell* [19].

This technique is more focused in avoiding faults instead of dealing with them. As a result circuit redundancy techniques are more often used being a more efficient alternative, as we will see in the next chapters.

### 3.2.2 Triple Modular Redundancy (TMR)

TMR is currently widely used. Its concept is straight forward and it is based mainly on spacial redundancy.

By multiplying a circuit element and introducing a voting mechanist we can theoretically suppress and correct any failure cause by soft-faults.

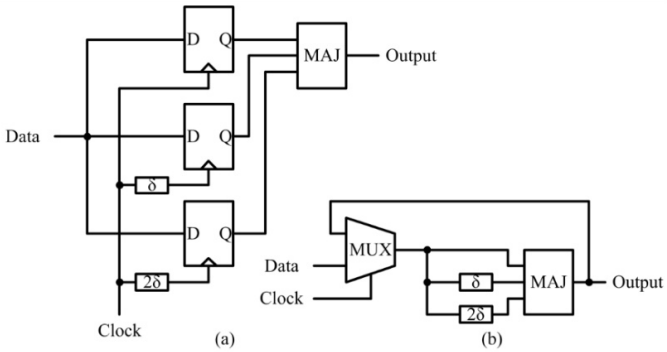


*Figure 4 - Triple modular redundancy latches from Mavis and Eaton [2002]. (a) Spatial sampling latch. (b) Temporal sampling latch. From* [20].

What is also obvious are the draw backs: at least 3x more power consumption and 3x more area cost[21].

There are also versions of this strategy that use temporal redundancy. This attenuates some of the drawbacks from the spacial solutions but imposes a bigger time penalty.

The main concept involves the triple processing of the same information that is stored and once again analyzed by a voting mechanism.

Figure 4 give us a visual example of a circuit implementing the two variations of TMR.

In addition, a software-based TMR solution is found in some papers[22]. A virtualization-based system would be set by several guests that can run virtually in the same processor but as if they were running in separated hardware. This solution, in the authors point of view, is a more architectural strategy but deserves to be mention here because of its similarity with the overall TMR concept.

### 3.2.3 Dual Modular Redundancy (DMR)

DMR tries to deal with the main disadvantages of a triplication of resources or time.

In some variants, the resilience of this solution reaches the levels of TMR.

This technique, requires specialized circuit design[21] because no reliable voting system is possible. The design usually gets outside the standard cell library typically provided by *Application Specific Integrated Circuit* (ASIC) and *Field Programmable Gate Array* (FPGA). That weights a lot in its adoption as a large scale solution.

Two variations of DMR are *C-Element Based Soft-Error-Resilient Latches* and *Dual-Interlocked Storage Cell* (DICE).

Both try to answer the question of how to identify an error between two outputs, something very straight forward in the case of TMR. The inability of identifying reliably the error also prevents mostly its correction.

The *C-Element* passes the value of its inputs to its output when both inputs are equal, or retains its previous output value[23]. This is managed by a *keeper*, protecting the system from propagating the error.

Some other variations use a master-slave strategy that simply uses the slave component only if the master is not available

In both cases, a SEMU can break the mitigation given by a DMR.

The mitigation of SEMU is nowadays in the forefront of research. Some solution were already proposed, as the next chapter describes.

### 3.2.4 Layout Design through Error-Aware Transistor Positioning (LEAP)

Circuit redundancy tends to be no longer an adequate solution since duplicated, or triplicated circuit components can reside in the same area of a high-energy impact. Instead of increasing the distance between redundant components, LEAP tries to improve the transistor placement to soft faults.

Most of the discussion moves around circuit topology and layout based on the knowledge of the charge collection process triggered by particle impacts.

The paper exposing this technique describes the following steps to produce a soft-error-resilient layout:

> *1. An analysis of the circuit response to a single event for each individual drain contact node in the layout, and*
>
> *2. A careful placement of each drain contact node in the layout based on the above analysis, such that multiple drain contact nodes act together to cancel (fully or partially) the overall effect of the single event on the circuit[24].*

The specificity of the solution exceeds the paper's reach. But it is obvious that the researchers have chosen a different approach when dealing with the problem. This solutions can easily be adopted by other techniques as the ones mentioned above.

An example given by the paper is the complementary SEMU resistance give by LEAP when implemented in combination with DICE; a solution with its strengths in mitigating SEU effects.

The use of circuit interactions and transistor placement to improve the soft error performance is supposedly a very cost effective solutions.

## 3.3 Architectural Techniques

Architectural techniques can be subdivided into micro- and macro-solutions[18].

The micro-solutions focus more on the flow of data, while macro-solutions touch on systems layouts and configurations. An interesting example is found in processor arrays that can deal with soft and hard faults.

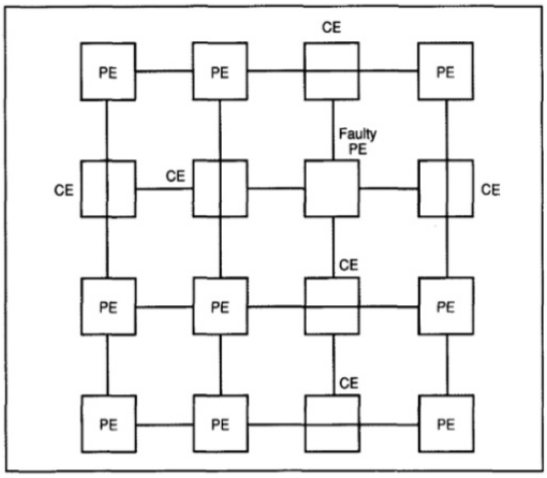


*Figure 5 – Row and column exclusion scheme for processor arrays.*

As the above Figure 5 shows, to a certain degree, a row and column exclusion scheme can recover from the total failure of a node. The node, instead of processing its input, merely passes the signal to the next node[7].

### 3.3.1 Error-Correction Code (ECCs)

Micro-solutions are based on ECCs or parity - a technique that checks whether data has been lost or written over when it is moved. They are a fine-grained solution that tries to control, at a bit level, the information being interchanged between some circuit

elements. This increases the design effort and also the complexity of an IC.

For example, inline ECCs requires a verification logic that checks the ECC code before returning to the next instruction. If the code underwent some change, the data must be discarded and the previous instruction replayed; this means one or more extra cycles in a processor pipeline.

Other solutions using parity error on an architectural register could be dealt by the operating system (OS), isolating the error to a specific process or set of processes. Killing the affected process or processes but leaving the rest of the system running is undoubtedly a pragmatic and cost-effective solution to soft faults[9].

Macro-solutions are based in mirror CPUs or threads. This is what we explore next.

### 3.3.2 Checkpoint and Recovery (CR) technique

CR is a temporal redundancy technique that does not need any modification in the software source code.

This technique is versatile in its implementation being a software-only or hardware-only solution. The principle is the same: creating specific and strategic checkpoints during a process where a state can be saved. If an error is detected, this enables the return to the previous valid state[25].

While software solutions are cheaper, the implementation of hardware for this purpose, lowers the overhead from this fault-tolerance solution.

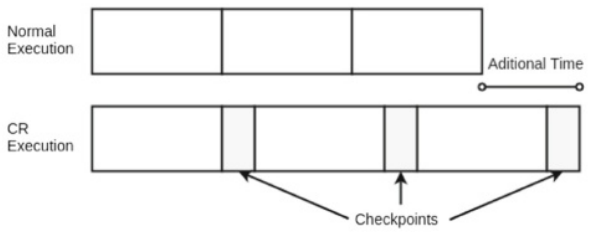


*Figure 6 – overhead when using CR[25].*

As we can see in Figure 6 this solution comes with the expense of processing time instead of hardware redundancy. The overhead comes from the time to check the results and in case of an error detection, the re processing of the last instructions.

### 3.3.3 Lockstep

Lockstep solution can be directly mapped to DMR or TMR; instead of a spacial redundancy by duplicating or triplicating hardware components, Lockstep runs processes in parallel, obtaining the same results as its *spacial* cousins. For that, a cycle-to-cycle check is done (similar to the CR technique), meaning a synchronization between all the running processes.

### 3.3.4 Dual-Core Lockstep - Lock-V

A very interesting article underlines the lack of literature discussing heterogeneous architectures, i.e. system that duplicate processor units but with different architectures like RISC-V and x86 [22].

Some faults can affect redundant components at the same time. This is known as *common-mode fault*. This can only be mitigated by adopting a more diverse system.

The Lock-V technique can be divided into two blocks. The first is the software block that uses a specific framework to add a checkpoint to the source code before compilation. This is needed because we must create two different binaries for two processor architectures.

The second block is the hardware block. Besides the processing system composed of the different processing units, there is also a denominated *xLockstep* accelerator that is used to sync both cores, evaluate the output, control the code execution when comparing results and even suspend the processors.

This hybrid solution (software and hardware) unveils some of the ground currently being explored in this field of research.

### 3.3.5 Redundant Multi-threading (RMT)

RMT can be implemented in either hardware or software because the checks can be done at the architecture level[18]. It is a technique also based on redundancy, as the name points out.

This solution is usually compared with Lockstep and uses a temporal strategy to deal with faults. The output of selected committed instructions is checked for errors. This does not necessitate synchronization between threads.

Less overhead during software design is one of the advantages underlined in some papers exposing RMT.

> *RMT) allows designers to reduce the number of software checks necessary compared to prior implementations of software fault detection[18].*

### 3.3.6 Duplication With Comparison (DWC)

This technique starts by duplicating the original circuit.

There is also the possibility of a partial duplication. However, the design must be fully duplicated to achieve full error coverage. This presents some issues like the duplications of inputs and outputs in buffers[26].

Next, and as the name indicates, a comparator is inserted in some locations off an IC. The comparator will compare the two outputs and identify possible errors.

This solution is similar to DMR but on an architectural level.

## 4. Conclusion

When talking about *fault avoidance* – in contrast to *fault tolerance* – we are talking about a more complementary and passive solution that answers some of the semiconductors industry demands but also the drawbacks coming from the continuous improvement in computation power.

In terms of fault tolerance, redundancy is the keyword in most solutions. Being temporal or spacial, redundancy implies an unoptimized use of resources – hardware, money, space, or time – when the only requirements are functional. Nevertheless, the natural world imposes other challenges.

Given the almost universality of redundancy, techniques like LEAP and Lock-V open new exciting possibilities currently being explored in industries like the automotive.

The crossing of heterogeneous data sources from sensors added to vehicles for decision-making in self-driving systems, are being used as a fallback in case one of the sensors is offline. The multiple *cameras*, *radars* and *LIDARs* are simultaneously improving the information gathering and the fault tolerance of the overall system.

> *These are not additionally redundant. The redundancy lies in the diversification of different sensors[27].*

This follows the pertinent theme that is raised in the Lock-V paper; heterogeneous systems are more resilient.

During the last two decades, when making a zoom in the development of self-driving cars, we can identify the main trends in the reinforcement of the overall system reliability in a static situation and, more recently, in a dynamic situations.

The state of the art for self-driving cars shows a combination of micro-controller units (MCU) supplied with independent sensors configured in a M out of N (MooN) redundancy. These MCUs are powered by independent power supplies and dedicated *watchdogs*[27].

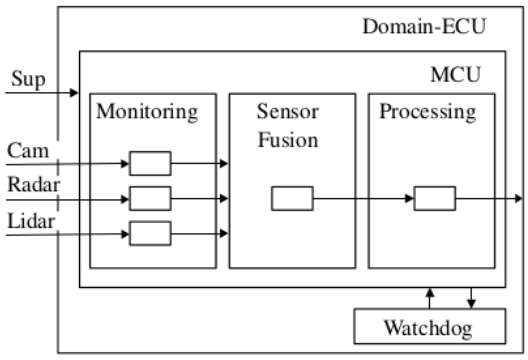


*Figure-7 - Structure of a MCU in 1oo1 (MooN) architecture.*

This design architecture uses an equilibrium between CPU and sensor number – here, the *MooN* expression – and hardware element reliability.

Modular components with single responsibilities, *watchdogs* that monitor the state of hardware and software with the capability of restarting them in case of failure[14] were always used in the industry. *Watchdogs* work here as the *xLocktep* in Lock-V.

The development of more resilient software and a fine-tuned relationship between sensors and CPUs are pushing a fully autonomous car closer to reality.

Right now, the main challenge is the latency from error recovery that, in a dynamic context like an urban area, is still too slow and unreliable to meet the security requirements.

By exploring this context, we can glimpse the probable trend in systems reliability: a mixture of circuit solutions and architectural solutions that, combined with system heterogeneity, can fulfill the most extreme requirements in terms of reliability.

-

# References


[1] Paulo R. C. Villa, Rodrigo Travessini, Roger C. Goerl, Fabian L. Vargas, Eduardo A. Bezerra "Fault Tolerant Soft-Core Processor Architecture Based on Temporal Redundancy", Springer Science+Business Media, LLC, part of Springer Nature 2019, February 2019, pp. 11.

[2] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, Setember 2011, pp. 39.

[3] Somashekhar, Dr.Vikas Maheshwari, Dr.R. P. Singh "A Study Of Fault Tolerance In High Speed VLSI Circuits", International Journal Of Scientific & Technology Research Volume 8, Issue 08, August 2019, pp. 1776.

[4] Constantinescu, Cristian "Trends and Challenges in VLSI Circuit Reliability", IEEE Micro, Volume: 23, Issue: 4, July-August 2003, pp. 15.

[5] Constantinescu, Cristian "Trends and Challenges in VLSI Circuit Reliability", IEEE Micro, Volume: 23, Issue: 4, July-August 2003, pp. 14.

[6] Puneet Gupta, Evanthia Papadopoulou "Yield Analysis and Optimization", Blaze DFM Inc. and IBM J Watson Research Center, November 2008, pp. 1.

[7] Shubhendu S. Mukherjee, Joel Emer, and Steven K. Reinhardt "The Soft Error Problem: An Architectural Perspective", Conference Paper, March 2005, pp. 4.

[8] Constantinescu, Cristian "Trends and Challenges in VLSI Circuit Reliability", IEEE Micro, Volume: 23, Issue: 4, July-August 2003, pp. 16.

[9] Shubhendu S. Mukherjee, Joel Emer, and Steven K. Reinhardt "The Soft Error Problem: An Architectural Perspective", Conference Paper, March 2005, pp. 2.

[10] Mohamed Mounir Mahmoud, Norhayati Soin, Hossam A. H. Fahmy "Design Framework to Overcome Aging Degradation of the 16 nm VLSI Technology Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 33, No. 5, May 2014, pp. 1.

[11] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, Setember 2011, pp. 6.

[12] Constantinescu, Cristian "Trends and Challenges in VLSI Circuit Reliability", IEEE Micro, Volume: 23, Issue: 4, July-August 2003, pp. 17.

[13] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, Setember 2011, pp. 2.

[14] T. M. Julitz, A. Tordeux and M. Löwer "Reliability of fault-tolerant system architectures for automated driving systems", Cornell University, October 2022, pp. 3.

[15] Podcast "Command Line Heroes" from RedHat, Episode "Robot as Vehicles", December 2021.

[16] Paulo R. C. Villa, Rodrigo Travessini, Roger C. Goerl, Fabian L. Vargas, Eduardo A. Bezerra "Fault Tolerant Soft-Core Processor Architecture Based on Temporal Redundancy", Springer Science+Business Media, LLC, part of Springer Nature 2019, February 2019, pp. 10.

[17] Constantinescu, Cristian "Trends and Challenges in VLSI Circuit Reliability", IEEE Micro, Volume: 23, Issue: 4, July-August 2003, pp. 18.

[18] Shubhendu S. Mukherjee, Joel Emer, and Steven K. Reinhardt "The Soft Error Problem: An Architectural Perspective", Conference Paper, March 2005, pp. 5.

[19] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, September 2011, pp. 25.

[20] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, September 2011, pp. 26.

[21] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, September 2011, pp. 27.

[22] Cristiano Rodrigues, Ivo Marques, Sandro Pinto, Tiago Gomes, and Adriano Tavares "Towards a Heterogeneous Fault-Tolerance Architecture based on Arm and RISC-V Processors", Centro ALGORITMI, University of Minho, October 2019, pp. 2.

[23] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, September 2011, pp. 28.

[24] Hsiao-heng Kelin Lee, "Circuit and Layout Techniques for soft-error-resilient digital CMOS Circuits", Dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University, September 2011, pp. 52.

[25] Paulo R. C. Villa, Rodrigo Travessini, Roger C. Goerl, Fabian L. Vargas, Eduardo A. Bezerra "Fault Tolerant Soft-Core Processor Architecture Based on Temporal Redundancy", Springer Science+Business Media, LLC, part of Springer Nature 2019, February 2019, pp. 12.

[26] Daniel L. McMurtrey "Using Duplication with Compare for On-line Detection in FPGA-based Designs", Brigham Young University, December 2006, pp. 28.

[27] T. M. Julitz, A. Tordeux and M. Löwer "Reliability of fault-tolerant system architectures for automated driving systems", Cornell University, October 2022, pp. 4.