Towards Single-Chip Diversity TMR for Automotive Applications

Omar Hiari, Waseem Sadeh, and Osamah Rawashdeh
Electrical and Computer Engineering Department
Oakland University
Rochester, Michigan
omhiari@oakland.edu, wasadeh@oakland.edu, rawashd2@oakland.edu

***Abstract*—The continuous requirement to provide safe, low-cost, compact systems makes applications such as automotive more prone to increasing types of faults. This may result in increased system failure rates if not addressed correctly. While some of the faults are not permanent in nature, they can lead to malfunctioning in complex circuits and/or software systems. Moreover, automotive applications have recently adopted the ISO26262 to provide a standard for defining functional safety. One of the recommended schemes to tolerate faults is Triple Modular Redundancy (TMR). However, traditional TMR designs typically consume too much space, power, and money all of which are undesirable for automotive. In addition, common mode faults have always been a concern in TMR which their effects would be increasing in compact systems. Errors such as noise and offset that impact a TMR sensor input can potentially cause common mode failures that lead to an entire system failure. In this paper, we introduce a new architecture and implementation for diverse TMR in a speed measurement system that would serve automotive cost and safety demands. Diversity TMR is achieved on a single chip by designing functionally identical circuits each in a different design domain to reduce the potential of common mode failures. Three versions of a speed sensing application are implemented on a mixed-signal Programmable System on Chip (PSoC) from Cypress Semiconductors. We introduce errors that impact speed sensor signals as defined by the ISO26262 standard to evaluate DTMR. Our testing shows how DTMR can be effective to different types of errors that impact speed sensor signals.**

Keywords-component; TMR; DTMR; SEUs; ISO26262; Functional Safety; Fault Tolerance;

I. INTRODUCTION

In recent times, Triple Modular Redundancy (TMR) has become one of the common effective error mitigation methods used to increase system reliability. TMR has been used in both aeronautic and ground systems. In aviation system applications, such as the Boeing 777, TMR is implemented across all systems to ensure reliable operation [16]. In addition, as automotive systems are moving towards safety critical x-by-wire systems, fault tolerant methods such as TMR are needed due to high reliability requirements. As a result, as higher standards for reliability become necessary, the cost and unit size still need to be kept at a minimum.

Two of the main constraints in automotive are cost and size. The nature of the automotive industry being focused on mass produced products makes it more cost driven. Size is also critical to reduce the overall vehicle weight. Nevertheless, TMR has traditionally been obtained by triplication of hardware functionality to reduce effects of common mode errors. Traditional approaches therefore are not the most cost or size effective for applications such as automotive. In addition, memory, power, and clock signals could be either shared or physically separate while using a single chip depending on how much separation is required. Therefore, SoCs provide a more cost effective platform on which TMR can be implemented. More of those implementations have to be studied, however, to understand how much of an advantage is gained over existing schemes.

As functional safety is becoming more prevalent in automotive applications, challenges are increasing to address the different environment effects to maintain highly reliable systems. Functional safety standards, such as the ISO 26262, have been developed to establish a baseline, which applications are required to meet. The reliability of an embedded system unit involves many different aspects such as; sensors and/or actuators, software, environment and EMC, integrated circuits, and verification and validation.

Automotive Safety Integrity Levels (ASILs) have been introduced as part of the recently released ISO26262 standard to evaluate the safety of automotive systems. System electronic modules are assigned an ASIL to indicate the level of safety a module can provide. ASIL D is the rating for the most dependable systems and ASIL A for the least dependable systems. Therefore to meet a certain ASIL level, a component design must incorporate all necessary fault detection and mitigation techniques required for achieving that level [15].

In the ISO26262 TMR, among other fault mitigation methods, is highlighted as one of the methods recommended to increase reliability of a system. TMR masks out faults by introducing three redundant copies of a system that are continuously voted on. As a result, TMR reduces the probability of having system failure if one of the blocks fails. However, given that circuits are identical, and TMR in its simplest form, does not have the capability to handle multiple failures, common mode faults remain to pose a special threat.

978-1-4673-0818-2/12/$31.00 ©2012 IEEE

Faults could be due to design faults in the redundant copies, Electromagnetic Interference (EMI), or even temperature. Depending on the faults of concern, some of the methods utilized in reducing common mode errors include using hardware manufactured by different suppliers, developing software for every block by different teams, or placing the different blocks physically far apart from each other. Existing methods therefore might increase the cost or size of systems required.

Recent advancements in system-on-chip solutions have allowed the design of mixed signal implementations on a single chip. System on Chip (SoC) designs can be flexible to allow good separation between designs as required while keeping the cost and size at a minimum. However, the reliability of such TMR designs needs to be further studied. Especially their tolerance to common mode faults.

We investigate the capabilities of a Programmable System on Chip (PSoC), with its diverse mixed signal resources, to provide lower cost TMR that is more immune to common mode faults. We use the Cypress PSoC3 platform to implement our work.

II. CONCEPT

The proposed scheme implemented to enhance tolerance of common mode faults is diversity triple-modular-redundancy (DTMR) [8]. As shown in Figure 1, the idea of traditional TMR is to increase system reliability by feeding three identical circuits into a "voter" which masks out the faulty circuit. The goal of TMR is to isolate the fault containment regions in between the different implementations. To maximize isolation of fault containment regions, traditional TMR methods usually call for the use of multi package solutions when it comes to silicon implementations. However, recent semiconductor trends have enabled the maximization of fault containment regions on a single chip where shared resources can be minimized.

TMR is categorized as an *M*-of-*N* system with a voter. An *M*-of-*N* system is one that consists of N redundant blocks that need at least M of them to be operational. Therefore, TMR would qualify as a 2-of-3 system. The reliability of any *M*-of-*N* system without a voter can be represented as [17]:

$$R_{M\_of\_N}(t) = \sum_{i=M}^{N} \binom{N}{i} R^i(t)[1-R(t)]^{N-i} \qquad (1)$$

Where *R(t)* is the reliability that presents the probability that a block is still operational at time *t* and,

$$\binom{N}{i} = \frac{N!}{(N-i)!i!} \qquad (2)$$

For a 2-of-3 system this reduces to

$$R_{2\_of\_3}(t) = 3R^2(t) - 2R^3(t) \qquad (3)$$

In the case the three blocks are affected by a common mode fault with a probability F_{CM} then the reliability becomes

$$R_{2\_of\_3}(t) = (1-F_{CM})(3R^2(t) - 2R^3(t)) \qquad (4)$$

Adding the reliability of a voter $R_{voter}(t)$ to represent a TMR system the expression becomes

$$R_{TMR}(t) = R_{voter}(t)(1-F_{CM})(3R^2(t) - 2R^3(t)) \qquad (5)$$

The focus of our work essentially is to keep the value of F_{CM} at a minimum in compact, low cost systems. Determining the value of F_{CM} nevertheless requires long hour testing that we preserve for future work. What will be proven on the other hand is that DTMR can potentially enhance the value of F_{CM} and thus qualify it for long hour testing. It can also be seen from the expression that the reliability of the voter is critical to proper operation of TMR as it is a single point of failure. However, reliability of the voter is not the focus of this work as there has been other work that addresses the reliability of voters.

Sources of common-mode failures are faults that cause redundant copies to fail under identical conditions. In the case of identical TMR systems, faults affecting the three system blocks could be identical. Therefore, for increased system reliability we explore the effect of diversifying the design of the TMR concept by implementing TMR in three different design versions: digital hardware, analog hardware, and software.

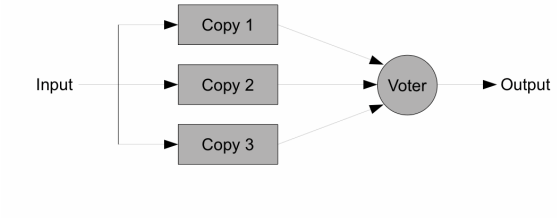

Figure 1 TMR Block Diagram

Table I demonstrates the characteristic differences between different design implementation domains. The main characteristic advantage that the analog design domain carries over the digital domain is it having continuous signal types. Continuous signal types have the advantage of preventing dramatic change in state. For example, in a digital system a direction variable can change from East to West due to a bit flip error. In an analog system, however, the change would be more gradual if an error occurs. On the other hand, digital domain designs for edge detect type sensors are less tolerant to higher voltage offsets than analog design especially in signals that require an edge detect (i.e., a tachometer type sensor signal in the case of our work). It must be noted however that constant voltage reading type sensor signals would have to be evaluated differently in offset cases.

In the case of random noise for analog designs, the quality of filtering would determine how well circuit performs. Generally it would be hard for a filter to keep out a large range of frequencies, therefore, the design could be more adversely affected. This assumes, however, that proper design guidelines have been followed. Digital domain

implementations, including software, can be affected equally by random noise where tachometer edges can be missed. In digital hardware and software implementations, however, as long as an edge is detected then random noise has virtually no effect throughout the remainder of the speed measurement operations. On the other hand, in an analog implementation the effect of noise can carry out through the whole circuit to produce a faulty output.

SEU immunity is critical as semiconductor packages keep on shrinking for digital devices. Therefore, as SEUs are more involved with bit flipping in memory then analog devices should be less prone to that effect if implemented external to the integrated device. SEUs are effectively what affects the remainder of speed measurement operations in digital domain operations. The rate at which SEUs occur is generally determined by what ASIL requirement level is needed for a system in automotive applications.

TABLE I. DESIGN DOMAIN CHARACTERISTIC COMPARISION

Characteristics	***Design Domain***		
	Analog Hardware	***Digital Hardware***	***Software***
Signal Type	Continous	Discrete	Discrete
Signal Monitoring	Continous	Continuous	Periodic
Immunity to Random Noise Coupled on Sensor Signal	Dependent on design and filter quality	Medium	Medium
Immunity to Voltage Offset Coupled on Sensor Signal	Design Dependent	Low for Edge Detect	Low for Edge Detect
SEU Immunity	High if implemented external to SoC	Low	Low

The last of the remaining characteristics is signal monitoring. Periodic type, interrupt driven monitoring in software implementations make it more likely for signals to be either missed or readings delayed. As a result, periodic type monitoring can potentially provide inaccurate readings. As demonstrated so far, collectively all the different characteristics of the different design implementations make it less likely for one type of error to affect all DTMR copies in a similar manner.

A Programmable System on Chip (PSoC) development board from Cypress Semiconductors is used for the D-TMR implementation of the speed sensing application. The PSoC provides a mixed signal platform that includes a CPU, a digital configurable block defined by a hardware description language, and an analog block allowing the containment of most of all three diverse design blocks using a single chip. In the analog version, minimal external components are still required given that the SoC analog implementation capability is limited.

The application chosen to implement our diversity TMR concept is a speed sensor monitor. We chose this application because sensor applications exist widely in automotive systems and are an essential part of almost any control system. Speed sensing applications are common in many safety critical embedded systems making the results of this study of wide interest.

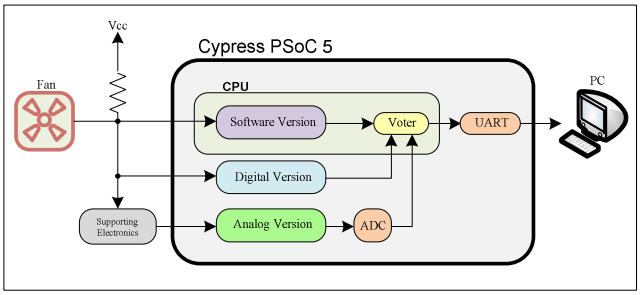

Figure 2 DTMR Block Diagram

III. IMPLEMENTATION

Our implementation consists of four main parts: the software version, the analog hardware version, the digital hardware version, and the voter. Figure 2 shows the entire block diagram implementation of the DTMR system. The three copies of the TMR structure all feed into a voter, which is implemented in software that produces the "voted" compare value to a UART block to feed the data to a PC that collects the data. A fan tachometer that produces 2 pulses per revolution is used to detect speed. The tachometer line feeds back to an analog frequency to voltage converter for the analog block, an interrupt pin for the software block and directly to the digital hardware block. The frequency to voltage converter provides a constant voltage that is proportional to the speed of the fan. The voltage is then scaled and fed into an ADC. The ADC output is directly fed into the voter for comparison with the other two copy outputs. More detail on the separate implementation versions is presented in the following subsections.

A. Software Version

The tachometer output, which is the current fan speed, is fed to an interrupt pin that is triggered at every rising edge of the tachometer signal. Another interrupt is also generated internally that triggers every rising edge of a 120kHz clock signal. The interrupt service routine saves the count and provides the period to the main loop. The main software loop then converts the count into an RPM value and stores it into a variable that is collected by the Voter algorithm.

B. Digital Hardware Version

As mentioned earlier, the PSoC platform allows the implementation of digital logic using hardware description language. Hence, the digital copy of our DTMR implementation is implemented in Verilog. As can be observed in Figure 3, the tachometer feedback signal is fed back directly to the digital block. The digital block maintains a count value that is incremented with every clock rising edge. In addition, at every rising edge of the tachometer

signal, the count value is stored to a different register and then reset. The stored count value is then used to calculate the RPM and stored into another register for Voter algorithm to retrieve.

C. *Analog Hardware Version*

The analog version is implemented using an external frequency to voltage converter and a voltage scaling circuit. Fully integrating the analog block can be achieved as PSoCs further develop their analog capability. For the mean time, an external frequency to voltage converter is needed to convert the tachometer signal to a constant voltage that is proportional to the period of the signal, and thus the speed of the motor. The F/V output voltage is then scaled appropriately ahead of feeding into the ADC so that an RPM value is available right after the conversion. The ADC output then stores the result to a software variable that makes the result available for the Voter algorithm to collect.

D. *Voter*

Similar to the work done by G. Borges *et al.*, the voter design was implemented in software because the majority of signals were already in digital format [1]. Therefore, only the analog block output had to be converted to a digital value. Due to the different execution times of the different implementations, the output signals of the blocks could potentially be out of sync. Therefore, a method had to be determined to take timing differences between the three copies into account. Voter synchronization techniques have been classified in the past into three types [12]:

1. Independent Accurate Time Bases: The blocks would synchronize for a short period of time. Then, the separate blocks would each rely on the accuracy of their own time base.
2. Common External Reference: The various blocks would share a common reference.
3. Mutual Feedback: The blocks have no common feedback and would have to synchronize with each other.

Given that there is not common time base between the three different versions it was decided to maintain synchronization by mutual feedback. In our implementation of mutual feedback, the voter synchronizes the conversions by maintaining two signals. The voter receives a READY signal from each block and in the same time feeds a common GO signal to each of them. The GO signal is triggered by the voter as a signal for every block to start its conversion and/or calculation. The READY signal provided by each block tells the voter that a value is available to be compared. Once all blocks have values available, the voter proceeds with the comparison algorithm to produce an output to the PWM block.

The voter is implemented in software because two of the blocks are already providing digital outputs. After receiving all the output values from the different blocks, the voter algorithm compares each of the received output values to each other. If a block output value drifts away from an acceptable range of the other two RPM signals, then it will be considered the faulty output and will be masked out. If there is no fault, then the voter output would default to one of the block outputs. Otherwise, if a fault occurs on one block only, a priority is given to the output that needs to be selected as default to pass on as an RPM value.

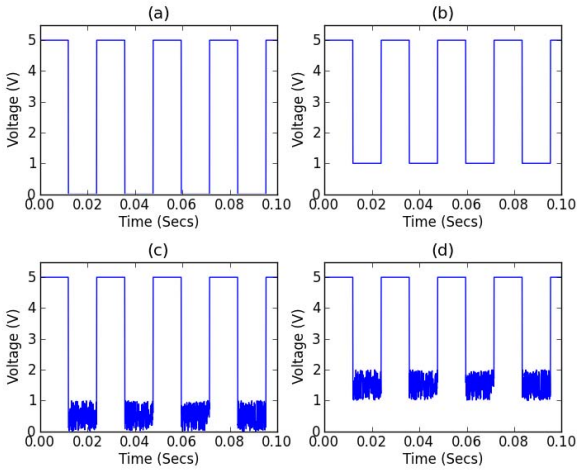

Figure 3 (a) Sensor Signal without faults (b) with 1V offset (c) with random noise (d) with random noise and 1V offset

IV. EXPERIMENTAL SETUP

In this work we consider failure modes of sensor elements as defined in the ISO26262 standard. There are 4 types of faults defined; out-of-range, stuck-in-range, oscillation (or noise), and offset. In our case oscillation and offset are the type of faults considered to be definite contributors to common mode failures. The out-of-range and stuck-in-range faults are not considered in this work because they are less likely to cause common mode failures and can be protected for by alternate methods. As a result, we will be evaluating the performance of the different DTMR blocks under oscillation and offset conditions.

We use another separate PSoC development board to inject the faults into the system. The PSoC platform has the capability to generate a random noise signal and/or a voltage offset. The generated signals are then coupled to the sensor output thus providing a "faulty" sensor output. In this evaluation the three different implementation outputs in addition to the voter output are checked separately and compared to each other.

Three types of faults are injected on the sensor signal separately for every test. The sensor output is a 0-5 V square wave signal running at a frequency reflecting the rpm of the fan (Figure 3a). The three types of tests are:

1. Offset Injection: A voltage offset from 0 to 3 V that increases linearly over the period of the test. The period of the test is selected to be 24 hours (1440 min). Figure 3b shows an example of the sensor signal with a 1 V offset added to it.

2. Noise Injection: 4 random noise signals with different maximum amplitudes are injected to the sensor signal. The maximum bandwidth of the noise signal is 100 kHz. The 4 different amplitudes considered are; 48 mV_{pp}, 240 mV_{pp}, 1 V_{pp}, and 2 V_{pp}. The test was run and failures monitored for duration of 240 minutes per each amplitude level. Figure 3c shows an example of the sensor signal with $1V_{pp}$ noise signal added o it.
3. Offset and Noise Injection: This test is a combination the previous two tests. The noise level is fixed at 1 Vpp and the offset is increased linearly over the 24h period of the test. Figure 3d shows an example of the sensor signal with offset and noise added to it.

V. TEST RESULTS

The entire speed range for the sensor is from 1000 rpm to 3000 rpm. In each of the test cases the fan speed is set to operate at 1250 rpm during the tests. The speed reading is considered to be out of range (failure occurred) if the reading exceeds ± 150 rpm of the required operating speed.

In the case of offset injection (Figure 4), the results showed that the analog implementation had the lowest failure rates until after 720 minutes. After 720 minutes the failure rate of the analog implementation exceeded the other two implementations. In addition, the test was stopped at 730 minutes because all of the implementations were reporting failures. The digital hardware and software implementations' failure rates showed a comparable gradual increase in failures as the offset increased over time. The voter output, however, reported fewer failures than two of the other implementations.

For the noise injection test, voter output failures were not reported until the noise maximum amplitude was increased to $1V_{pp}$. Figure 5 shows the number of failures reported over the period of the test for every noise amplitude level. In this test also the analog implementation reported fewer failures than the other two implementations as the noise level increased. The voter output failure rate was constantly less than two of the other implementations.

In the last test a similar result to the first case of offset injection was observed. The main difference is that the test had to be stopped at 450 minutes due to all implementations reporting failures. Figure 6 shows the results of the commenced test. It can be notices that there are larger jumps in failure rates over the duration of the test. That is due to the randomness in the amplitude of the injected noise signal.

For the type of faults injected in this work the analog hardware implementation has shown the lowest failure rate followed by the digital hardware and then the software implementation. We note from the results that the voter output has always shown a better failure rate than the software and digital hardware implementations. The better failure rate of the voter output is driven by the low failure rate of the analog implementation. As a result, it can be seen that if the software or digital implementations were the only ones used in the TMR implementation then the possibility of common mode failures occurring would have been much higher in all three tests.

VI. RELATED WORK

In this work we adopt a Diversity TMR concept similar to the one presented by G. Borges *et al.* [1]-[3]. They have implemented a low-pass filter using the D-TMR concept on a PSoC platform. The filter circuitry was completely implemented on one chip. However, true separation was not achieved due to certain peripherals being shared between the different copies.. In addition, the voter was implemented in software and compared only the differences in amplitude between the output signals passing through the filter of each copy. For fault injection, the work only considered SEU type faults. We build on this concept by implementing a new architecture for a DTMR speed monitoring system that provides a better platform for evaluating the reliability of the implementation and considers further common mode faults.

As opposed to filters a constant output measurement, similar to the one in this work, would result in a better platform for reliability assessment. Filters impose more of a difficulty for voter implementations given that the gain of the filter might not match over the whole frequency

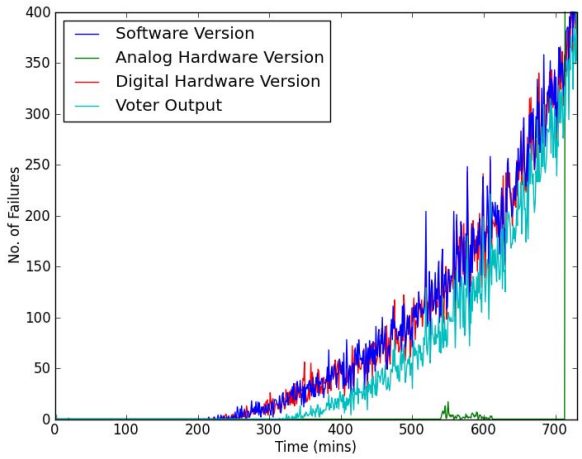

Figure 4 Offset Injection Test Results

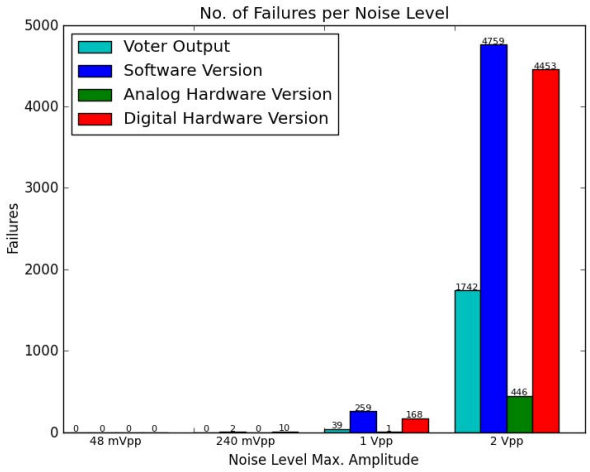

Figure 5 Noise Injection Test Results

response of each filter copy. As a result, in a filter DTMR implementation some true failures would not be detected.

Other work has addressed TMR in SoCs by implementing multiple processor SoC designs that can tolerate faults by using a low-level voting scheme [7]. Some others have addressed the problem by using multiple microprocessors [5,6]. In contrast, we concentrate more on replicating functions for SoCs with a single core rather than multiple cores. This would result in more compact implementations that are more cost effective.

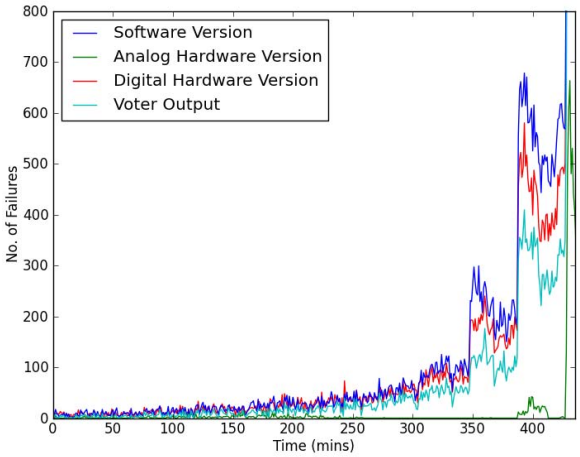

Figure 6 Offset Plus Noise Injection Test Results

VII. FUTURE WORK

In this work so far, we have provided a new architecture and implementation for a DTMR system that would serve automotive cost demands. The next step in our research is to continue the fault injection campaign by additionally introducing SEU type errors and commencing long hour tests. We are currently working on creating a test bed for injecting multiple type faults into a DTMR implementation to evaluate its reliability over long hour operation. Following the completion of the injection campaign, we will be comparing the difference in reliability achieved by DTMR implementation vs. traditional TMR implementations. As a result, we will also determine the types of errors in SoC applications where DTMR would be most effective in mitigating.

VIII. CONCLUSION

This paper presented an evaluation of a new method that can address the problem common mode faults in automotive embedded systems in a more cost, power, and size effective manner. The implementation of a new DTMR architecture for a speed monitoring application for automotive embedded systems was described. From the initial results shown DTMR should increase the reliability of systems by reducing the possibility of common mode failures due external faults imposed on the sensor signal. DTMR provides an alternate method to increase reliability at minimum added system cost.

ACKNOLEDGEMENT

This research is supported in part by the Michigan Space Grant Consortium (MSGC). The authors would like to also thank our lab members, especially Dr. Belal Sababha, for their insights and help in this project.

REFERENCES

[1] Gabriel de M. Borges, Luiz F. Gonçalves, Tiago R. Balen, Marcelo Lubaszewski, 2010, "Evaluating the Effectiveness of a Mixed-Signal TMR Scheme Based on Design Diversity", SBCCI '10: Proceedings of the 23rd symposium on Integrated circuits and system design, Pages: 134-139.

[2] Gabriel de M. Borges, Luiz F. Gonçalves, Tiago R. Balen, Marcelo Lubaszewski, 2010, "Diversity TMR: Proof of concept in a mixed-signal case", Latin American Test Workshop (LATW), Page(s): 1-6.

[3] Gabriel de M. Borges, Luiz F. Gonçalves, Tiago R. Balen, Marcelo Lubaszewski, 2010, "Increasing reliability of programmable mixed-signal systems by applying design diversity redundancy", 15th IEEE European Test Symposium (ETS) , Page(s): 261.

[4] A. Manzone, A. Pincetti, D. De Costantini, 2001, "Fault tolerant automotive systems: an overview", Proceedings of the Seventh International On-Line Testing Workshop, Page(s): 117-121.

[5] A. Bertacchini, P. Pavan, L. Tamagnini, M. Mistrorigo, M. Morandi, 2006, "Hardware-in-the-Loop Approach for Redundant Brushless Motor Control System", IEEE 32nd Annual Conference on Industrial Electronics, Page(s): 4054- 4059.

[6] A. Bertacchini, P. Pavan, L. Tamagnini, L. Fergnani, 2005, "Control of brushless motor with hybrid redundancy for force feedback in steer-by-wire applications", 31st Annual Conference of Industrial Electronics Society, Page(s): 1407-1412.

[7] Emmanuel Touloupis, James Flint, Vassilios Chouliaras, David D. Ward, 2005, "A Fault-Tolerant Processor Core Architecture for Safety-Critical Automotive Applications", SAE 2005 World Congress & Exhibition.

[8] Stephen Y. H. Su, Richard J. Spillman, 1977, "An overview of fault-tolerant digital system architecture", Proceedings of the national computer conference, Page(s): 19-26.

[9] Francis P. Mathur, Algirdas Avižienis, 1970, "Reliability analysis and architecture of a hybrid-redundant digital system: generalized triple modular redundancy with self-repair", Proceedings of the spring joint computer conference, Pages: 375-383.

[10] E. Normand, 1996, "Single Event Upset at Ground Level", IEEE Transactions on Nuclear Science, Volume: 43 , Issue: 6, Part: 1, Page(s): 2742 – 2750.

[11] R. Baumann, 2005, "Soft errors in advanced computer systems", IEEE Journal on Design & Test of Computers, Volume: 22 , Issue: 3, Page(s): 258 – 266.

[12] D. Davies, J.F. Wakerly, 1978, "Synchronization and Matching in Redundant Systems", IEEE Transactions on Computers, Volume: C-27 , Issue: 6, Page(s): 531 – 539.

[13] N. Seifert, Xiaowei Zhu, L.W. Massengill, Dec 2002, "Impact of scaling on soft-error rates in commercial microprocessors", IEEE Transactions on Nuclear Science, 49(6), Page(s): 3100–3106.

[14] Nathan Rollins, Michael J. Wirthlin, Michael Caffrey, Paul Graham, 2003, "Evaluating TMR Techniques in the Presence of Single Event Upsets", International Conference on Military and Aerospace Programmable Logic Devices.

[15] W. Gulland, 2004, "Methods of Determining Safety Integrity Levels", 12th Safety-critical Systems Symposium.

[16] Y.C. Yeh, 1996, "Triple-triple Redundant 777 Primary Flight Computer", Proceedings of the aerospace application conference, Volume: 1, Page(s): 293 – 307.

[17] Israel Koren, C. Mani Krishna, Fault Tolerant Systems, 1st ed. , vol. 1. Morgan Kaufmann, 2007, pp. 20 – 23.