EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESING (OBDP2021), 14-17 JUNE 2021

IP TO DETECT AND DIAGNOSE ERRORS IN COTS MICROPROCESSORS THROUGH THE TRACE INTERFACE

M. Peña-Fernandez[(1)], A. Lindoso[(2)], L. Entrena[(2)]

[(1)] Arquimea, 28918, Leganés, Spain
[(2)] Department of Electronic Technology, Universidad Carlos III de Madrid, 28911, Leganés, Spain

ABSTRACT

This work presents an error detection and diagnosis IP for space applications to enable fault tolerance by error detection and recovery on COTS processors. Its low-latency error detection capabilities and richness of trace information allow to perform effective fault diagnosis.

1. INTRODUCTION

Microprocessors are commonly used in all kinds of applications, such as commercial appliances, industrial controllers, communications, and embedded systems. They are becoming more common also in safety-critical applications. When operating in harsh environments, as in presence of radiation, microprocessors can be affected by faults, which may alter their intended behavior producing undesirable errors [1].

Nowadays, there are diverse techniques to design or build integrated circuits and microprocessors to be intrinsically resilient to radiation-induced errors, as Radiation Hardening By Design (RHBD) or Radiation Hardening By Process (RHBP). However, these hardening techniques usually lead to expensive solutions which cannot be afforded in cost-constrained applications. Moreover, by applying such techniques, resulting systems commonly require higher power and provide lower performance than commercial counterparts. In fact, the available rad-hard integrated circuits lag two or more generations behind commercial equivalent components [2].

In the last years, the interest in Commercial Off-The-Shelf (COTS) components for safety critical and even for space applications has increased, as they are attractive due to their higher performance and lower power consumption compared to their hardened counterparts. Nevertheless, when using COTS components, is the task of the system designer to assess the fault tolerance capabilities are at an acceptable level, transforming the traditional risk avoidance paradigm into risk management [3].

The use of COTS cutting-edge processing systems in space applications has received much attention due to an increasingly competitive commercial space sector. Such components would increment the processing capabilities on orbit to unprecedented levels, bringing a great competitive advantage. However, assuring reliability under the harsh space conditions is a challenge [4].

Single-event effects (SEEs) are a major concern in processors [5]. When using COTS components, available SEE protections are limited and the knowledge about the behavior of the device under radiation is poor. Error detection and diagnosis in modern microprocessors is a challenge, particularly due to the limited observability of the microprocessor internal resources. Typically, limited actions can be performed on the hardware to enhance radiation hardness. For that reason, COTS processors usually introduce software-level hardening, by modifying the code to increment robustness, but paying significant performance penalties. Moreover, software hardening can only protect software-accessible resources, but other processor resources may be left unprotected [6].

To achieve fault tolerance, processing systems based on COTS must be designed to implement error detection and recovery capabilities. However, few details are typically available about the internal architecture or implementation of COTS components, and the observability of the processor internal state is usually low. In addition, different failure modes presented by complex processing systems may need for diverse mitigation strategies, especially when considering different criticality levels [7].

Providing new solutions for COTS processors hardening may expand their usage in space missions and the associated on-board processing capabilities. In addition, subsequent cost and time reductions would make the space a more accessible market.

In this work we present an IP module to detect and diagnose errors in microprocessors working under radiation environments. The IP uses the information provided by the trace interface of the processor to check execution flow and data correctness with low latency and no performance penalty. The IP has been developed within an industry-academia collaboration as part of a Ph.D. thesis and is currently available at Arquimea.

The paper is organized as follows. Section 2 summarizes the related work in the field. Section 3 describes the proposed technique. Section 4 illustrates

some application cases. Finally, section 5 presents the conclusion of this work.

2. MICROPROCESSOR ERROR DETECTION AND DIAGNOSIS

Hardening techniques for microprocessors are usually classified into hardware or software techniques. Hybrid hardening techniques are considered when both hardware and software are addressed to harden the device [5]. Such techniques are designed to address errors that can be classified into two main categories: those affecting execution flow, called control-flow errors, and those only affecting program data, called data errors.

Software data hardening techniques usually rely on data replication and performing the same computations independently in each set of replicated data. The results of different computed data sets are then compared to stablish whether an error is present or not [8]. Data duplication techniques can effectively detect errors although data triplication is needed to perform error correction. As more data is replicated, associated performance and memory penalties are more severe, so tradeoffs must be considered to optimize error coverage against performance [9].

Software control-flow hardening techniques rely on signatures or assertions [10] to detect incorrect jumps in execution. Signatures are invariant results that are computed and checked during execution time. Assertions are special statements inserted in the application code to check the correctness of the executed code [5]. Extensive computation and checking of signatures and assertions may introduce significant performance and memory overheads.

A common problem for all software approaches is that their coverage is limited to the resources which are accessible from software. Internal microprocessor resources, such as the pipeline registers, can exhibit critical fault modes which may be left unprotected by software techniques [6].

Hardware techniques commonly modify the architecture to introduce redundancy, being Triple Modular Redundancy (TMR) the most representative case. However, in COTS it is not possible to replicate or even to modify the hardware. As an alternative, external hardware can be used to determine the correctness of the processor behavior. The complexity and operation of the external hardware ranges from simple watchdog timers to bigger modules that may become as complex as the observed processor. Complex observer modules may increase power consumption and area requirements and may also introduce new faults on the system [11]. Besides, the connection point for the external hardware is a critical issue that has impacts on performance, error detection latency and observability limitations.

An alternative approach for hardware redundancy is the replication of the entire processor, having two or more processors within a system. As processors become increasingly affordable, designers are leveraging the increasing availability of multicore processors in a single chip. Several approaches based on COTS multicore processors have been proposed for space applications, given that the faults on one processor core can be isolated from the other cores [12]. Lockstep is an extension of processor replication in which the execution of the replicas is synchronized. Simultaneous and symmetrical execution of the same application code should provide identical results in the absence of errors. If results differ, a rollback mechanism is needed to restore the system back to an error-free state. This approach is effective to detect data errors by comparing data results at several checkpoints during execution. However, in the case of control-flow errors, any of the processors could miss a checkpoint, resulting in an unprotected hang of the system. A watchdog timer could be used to overcome this limitation, but the associated high latency limits the efficiency of this solution. In addition, control-flow errors can become very complex, leading to latent effects that may not be reverted by system restoration [13].

Radiation testing is the most widely accepted method to evaluate the suitability of electronic devices for space applications, including data processing systems. Radiation testing results may quantify the device susceptibility to errors and the ability of hardening techniques to detect them and/or mitigate their effects. However, common testing approaches do not generally pay special attention to the causes of such errors and the associated circuit vulnerabilities. By increasing the knowledge related to the sources of errors, it could be possible to protect the circuits in a more effective manner and improve mitigation techniques. Additionally, gaining insight about the faults can lead to assess the criticality of an error, i.e. risk management [3], to take the corresponding corrective action.

Most existing diagnosis approaches evaluate the effects of errors and then try to deduct the origin of the fault using cause-effect analysis. However, the knowledge about the internal architecture and the observability of the system are crucial factors to effectively diagnose the error. A systematic approach is to evaluate the Architectural Vulnerability Factor (AVF) [14] of each processor resource to estimate their susceptibility to errors. Another approach is to perform extensive fault injection campaigns to create a fault dictionary associating fault location and observed effects to diagnose radiation-induced errors [15]. However, radiation-induced errors may present different characteristics from the modelled ones, limiting the effectiveness of such techniques. Moreover, fault consequences deeply depend on the application in execution, so it is difficult to develop a generic association between the errors and their origin. In addition, there are common errors, such as processor

hangs or crashes, that may have diverse causes, increasing the complexity of the diagnosis task regardless of the fault diagnosis approach. The accuracy of fault diagnosis strongly depends on the quality and completeness of the gathered information about the error. Collecting the information immediately after the event is crucial to avoid losing relevant data that could be overwritten.

The trace interface is a resource commonly found in modern microprocessors to support application development. It is initially intended to support software debugging and application profiling, by capturing relevant information concerning processor execution flow and data for those purposes. Such information is provided with low latency in a non-intrusive manner. However, once the application development is complete, the trace interface is commonly unused, so it can be reused for a different purpose with no cost.

Trace information is best suited for dealing with asynchronous events, such as those produced by radiation. However, the use for error detection and diagnosis is new and it is not natively supported by the processor manufacturers or associated tools. In addition, the use of computer-based tools may not be suitable for detecting errors in an embedded system while it is in operation. For this reason, a special infrastructure must be developed to leverage the information available at the trace interface for error detection and diagnosis.

The use of the trace infrastructures for processor online monitoring was first proposed in [16] to observe the execution of a LEON3 microprocessor and detect faults by computing signatures and comparing executions. Later works on this topic focused on soft-core microprocessors, which can be conveniently adapted or modified as needed to provide a wide and rich access to trace information [17]. However, the case of hard-core microprocessors is different. As the hardware cannot be modified and the internal resources cannot be accessed, the trace information must be obtained through hard macrocells that impose protocols and limit the available information [18].

ARM processors have achieved large market share in the commercial sector from the last two decades, and ARM-based space-oriented initiatives, such as Nanoxplore FPGAs or NASA HPSC, are becoming common. A wide range of competitive processor cores optimized for diverse applications, from low power to high performance, along with the ease of implementation in a System-on-Chip (SoC) may be two key factors for its success. ARM processor cores are also widely supported by software developers and libraries in many application fields, providing a huge knowledge base for new developments. CoreSight [19] technology is a family of components provided by ARM to support trace and debug capabilities on its processor cores. Almost every available ARM processor core is compatible with CoreSight technology.

3. ERROR DETECTION AND DIAGNOSIS IP

We are presenting a solution to tackle both radiation hardening and testability challenges regarding COTS microprocessors. We have developed a lightweight IP core in HDL that leverages the information available at the trace interface to detect and diagnose errors in ARM microprocessors, although the same approach could be applied to other processor architectures. The presented IP can oversee the behavior of a microprocessor or a SoC including more than one processor core, which is labeled as Processor Under Monitoring (PUM) within this document. The IP can observe execution flow and data values of PUM by monitoring the information provided by the trace interface in real time. It gives to the user the capability of detecting errors and obtaining error evidence and traceability with low latency, low impact on system design and no performance penalty.

The IP is currently compatible with several ARM CoreSight trace components: Program Trace Macrocell (PTM), Instrumentation Trace Macrocell (ITM), Trace Funnel and Trace Port Interface Unit (TPIU) [19]. The IP is compatible with the trace interface protocol specification, attending specifically to the trace information that can be used to detect errors. To that end, the IP is designed to obtain Program Counter (PC) values and data values from trace data. The IP has been designed to require low power and small area to be embedded in an application with minimum penalties. Regarding performance, the IP design is optimized to decode and process trace data in real time to minimize fault detection latency. The implementation of the IP can be adapted to multiple scenarios thanks to its parametric design. Low pin count interface enables multiple integration schemes. It can be used as a microprocessor peripheral on a System on Chip, or as standalone in a multi-chip system.

The IP has been developed and tested using Xilinx Zynq-7000 APSoC [20], integrating a dual core ARM Cortex-A9 processor.

3.1. Interface description

The IP can be connected to other devices through a set of interfaces, each one with a specific purpose within the intended error detection and diagnosis functionality. The top-level of the IP architecture is depicted in Fig. 1.

Configuration interface. The IP is configurable through a set of configuration registers that can be accessed via the following compatible configuration interface options.

- Advanced eXtensible Interface (AXI), for memory mapped SoC integration.
- 4-pin Serial Peripheral Interface (SPI), for multi-chip integration.

The configuration interface also provides access to information related to error diagnosis.

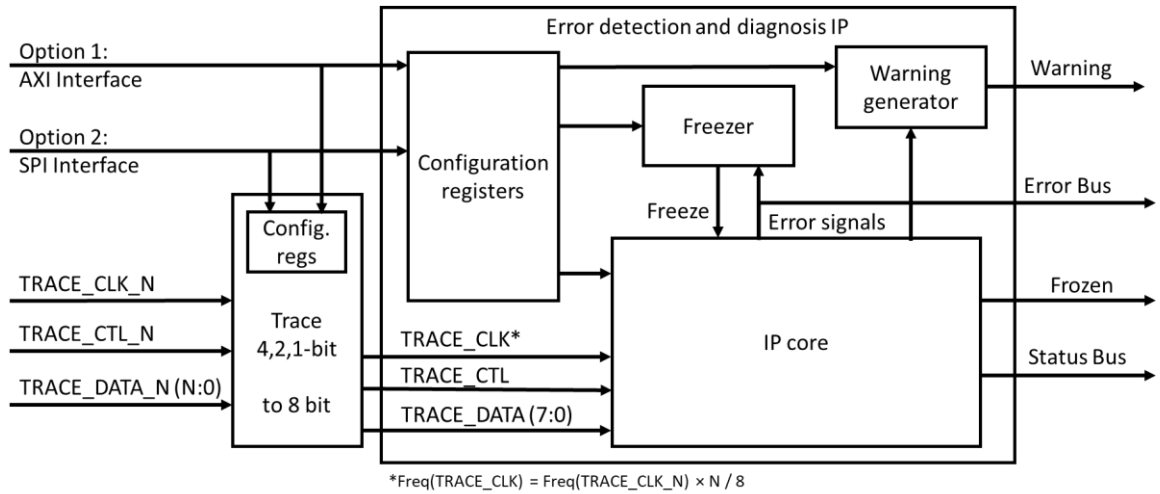

Figure 1. Top level view of the IP and interfaces

Trace interface. The IP trace interface is pin-to-pin compatible with ARM Trace Port Interface Unit (TPIU) pinout, which is present in most ARM processor implementations. The following signals are used:

- TRACE_CLK: clock signal to synchronize trace data.
- TRACE_CTL: control signal to indicate whether trace data is valid or not.
- TRACE_DATA (N:0): variable bit width trace data stream.

The IP is designed with 8-bit trace data port width by default. To enable compatibility with 1-bit, 2-bit and 4-bit trace data port widths, an available additional module must be inserted between the trace port and the IP.

Warning signal. Warning generator module can be configured to produce a warning signal upon the activation of any user-selected signals at error bus. This is typically used to indicate that an error has appeared but that the application can continue running, for example a data corruption that does not need for system reset, but only to ignore recently computed data.

Frozen signal. Freezer module can be configured to freeze the entire IP core upon the activation of any user-selected signals at error bus. Freeze signal is the resulting OR operation among all user-selected error signals at error bus. Once Freeze signal is activated, Frozen signal activates to indicate this situation. Once the IP is frozen, no further trace data will be processed, preserving the IP state for the user to gather error information through the configuration interface. In such a case, both PUM and the IP must be put back into a working, known state before continuing the application.

Status Bus. Information about the state of the internal resources of the IP is provided in this bus.

Error Bus. Every error signal generated by any internal resource of the IP is provided in this bus.

3.2. Functional description

The core of the IP is responsible of the management of the trace data supplied by the PUM, which is handled by a sequence of modules as depicted in Fig. 2. The IP works according to the following flow:

1. Trace information is generated on the PUM (Processor Under Monitoring) and exported to the IP through the TPIU. Inside the IP, it first enters the Reformatter, which decodes formatted trace frames and rebuilds the original trace stream from each source.
2. Depending on the source the trace comes from, it is sent to the corresponding trace decoder by the ID demux, which can be configured by the user with the identification code (ID) corresponding to each trace source present in the PUM.
   - The ITM decoder implemented in the IP can decode trace information produced by an Instrumentation Trace Macrocell, and the value retriever module obtains the values sent through the trace. Obtained data values can be sent to different user-configurable data checking resources, explained in section 3.3.
   - The IP can include one or more PTM decoder modules, each decoding trace information produced by a Program Trace Macrocell. The PC follower module obtains traced PC values, which correspond to a succession of instruction addresses of the corresponding PUM processor core. PTM decoder modules do not need a copy of the executed program to work. Thus, user is encouraged to enable branch broadcasting feature on PTM to maximize PC observability. Obtained PC values are sent to a set of checking resources, discussed in section 3.3.
3. Checking resources examine the information received from the trace interface and, according to

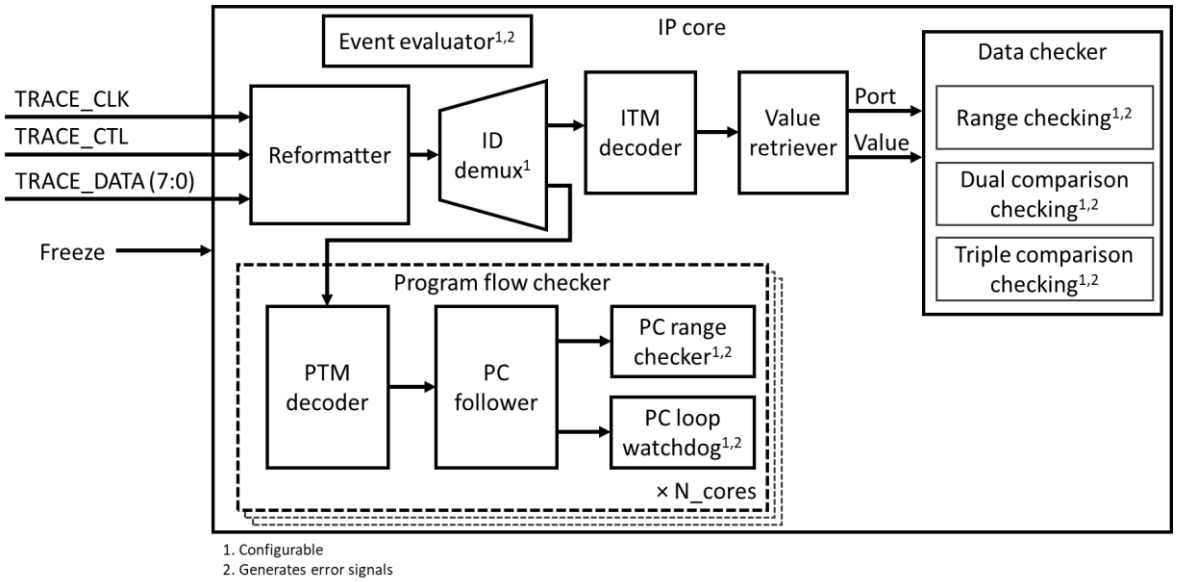

Figure 2. Internal architecture view of the IP

their configuration, raise a dedicated error signal upon an error.

4. The event evaluator module can perform logic operations with error signals to generate further error signals depending on more complex conditions.
5. If Freeze signal is activated at any time, all IP core resources become frozen, preserving their state to enable error information retrieval by the user.

3.3. Checking resources

The error detection capabilities of the IP are defined by the integrated checking resources, represented in Fig. 2.

Data checking. Different data checking resources are available, as data range checking, data dual comparison checking and data triple comparison checking. When the data value enters the data checker, it is sent to each resource according to user configuration. The same data value can be sent simultaneously to more than one resource:

- Data range checking resource generates an error signal whenever the received value is outside the expected user-configurable bounds. User can configure this resource to change the behavior and produce an error if the value is inside bounds.
- Data comparison checking resource can be dual or triple and generates an error signal whenever the received values match the corresponding Boolean operator. Both dual or triple type and Boolean operator are defined at implementation. User can also configure this resource to produce an error whenever the received values do not match the corresponding operator. Data is received sequentially, and the comparison is only performed when the last data value is received. For that reason, a configurable watchdog timer module is included in comparison checking resources to detect when a group remains incomplete for an excessive time.

Program flow checking. PC range checker and PC loop watchdog resources receive the successive PC values from a single PUM core to check whether the execution flow is correct or not.

- PC range checker resource constantly monitors all received PC values and raise an error signal in the case a particular value is outside of a set of user-configurable allowed ranges.
- PC loop watchdog resource is also constantly monitoring received PC values to check that a maximum time is elapsed between two consecutive receptions of a specific PC value. Selected PC value is commonly the first instruction of the main loop. In that case, the watchdog can be configured by the user to accurately detect functional interrupt errors by configuring the watchdog to raise an error signal in the case the elapsed time is greater than the maximum expected main loop execution time. Unlike traditional watchdog approaches, that rely on the processor to refresh the watchdog timer value, this approach does not need any action from the processor, improving reliability.

Combined resources checking. The IP handles trace data from different sources simultaneously. For this reason, additional checking approaches can be designed to integrate information from more than one resource to detect and diagnose errors. These features are currently under development and will appear in the next release of the IP.

- A lockstep checker could be implemented by combining PC information from more than one core running the same application in lockstep. Lockstep integrity could be checked by the IP in a non-intrusive manner and with no performance penalty.

Table 1. IP specifications

	Condition	Min	Typ	Max	Units	Comment
Pin count	SPI interface option No error signals	6	10			Each error signal adds extra pins
Error detection latency	No nested events in event evaluator			23	TRACE_CLK clock cycles	Event evaluator adds one cycle per each nested event
		140			ns	
Operating frequency	Implemented on Xilinx XC7Z010			166	MHz	TRACE_CLK frequency
LUT count	Synthesis for Xilinx Artix 7 series	2500	6000			6-input LUTs
Flip Flop count	Synthesis for Xilinx Artix 7 series	2700	7000			D-type FFs
Trace Data throughput	On-chip XC7Z010 over EMIO 8-bit data width			1333	Mbps	
	Off-chip XC7Z010 over MIO LVCMOS33 4-bit data width			920	Mbps	
	Off-chip XC7Z010 over EMIO TDMS33 4-bit data width			1200	Mbps	

- Signature/assertion checking can also be achieved by combining PC information with data values from the trace. This way, the IP could check the correctness of the execution flow not only by checking the PC value against allowed ranges, but also by checking the correctness of the associated signature values online with execution.

3.4. Error diagnosis

The information available at the trace interface is very rich, and the data rate for a typical application can exceed 1Gbps. The IP is an independent entity designed to decode and examine such huge amounts of data to detect errors. But the IP not only obtains error-related data, but also trace data related to nominal execution. Thus, it is possible to go a step further by using such information to contextualize error appearance. If only error detection is performed from trace data, most relevant information about the error would be lost. However, by gathering such information, a wider view of each error can be obtained, and error diagnosis can be achieved. For example, when a faulty PC value is found, the user could get the previous PC values, that would give the point in execution where the error took place. In addition, when a faulty data value is found, the user could also observe the faulty value and previous ones.

The presented IP has been designed to provide error diagnosis capabilities, by introducing historical data record on each checking resource. Once the IP has detected an error, it becomes frozen for the user to retrieve such historical information through the configuration interface.

4. APPLICATIONS

The IP has been developed in HDL, ready to be implemented in any FPGA platform. The parametric design of the IP increases flexibility and provides a wide range of user-configurable resources. Additionally, pre-implemented ready-to-use typical use case designs have been developed and can be provided to be used in commonly available development platforms for a quicker setup, evaluation, and deployment. Main IP specifications are listed in Tab. 1. The IP features high data throughput with small footprint, reduced pin count, and low latency.

Several development phases are supported by the provided functionality:

- **Design:** providing error detection and diagnosis capabilities during development to identify flaws in the system and enhance a given application to meet dependability requirements.
- **Device evaluation:** detecting and classifying errors in different devices, allowing severity evaluation to provide objective criteria on component selection. Not only for COTS but also for space-oriented devices, it could help to understand and mitigate complex failure modes.
- **Operation:** working side by side with a microprocessor to check the integrity of the executed application in real time, raise an alert upon error, and provide diagnosis information to perform the necessary corrective action with low latency, achieving fault tolerance.

The IP can be integrated using two basic system architectures: binary architecture or ternary architecture, depending on the number of available processors in the system.

In a **binary architecture**, only one processor, PUM (Processor Under Monitoring), and one IP are present. The IP is checking PUM execution through its trace interface and reporting error detection and diagnosis information to take corrective actions. If the found error is not recoverable, the IP would trigger a whole system

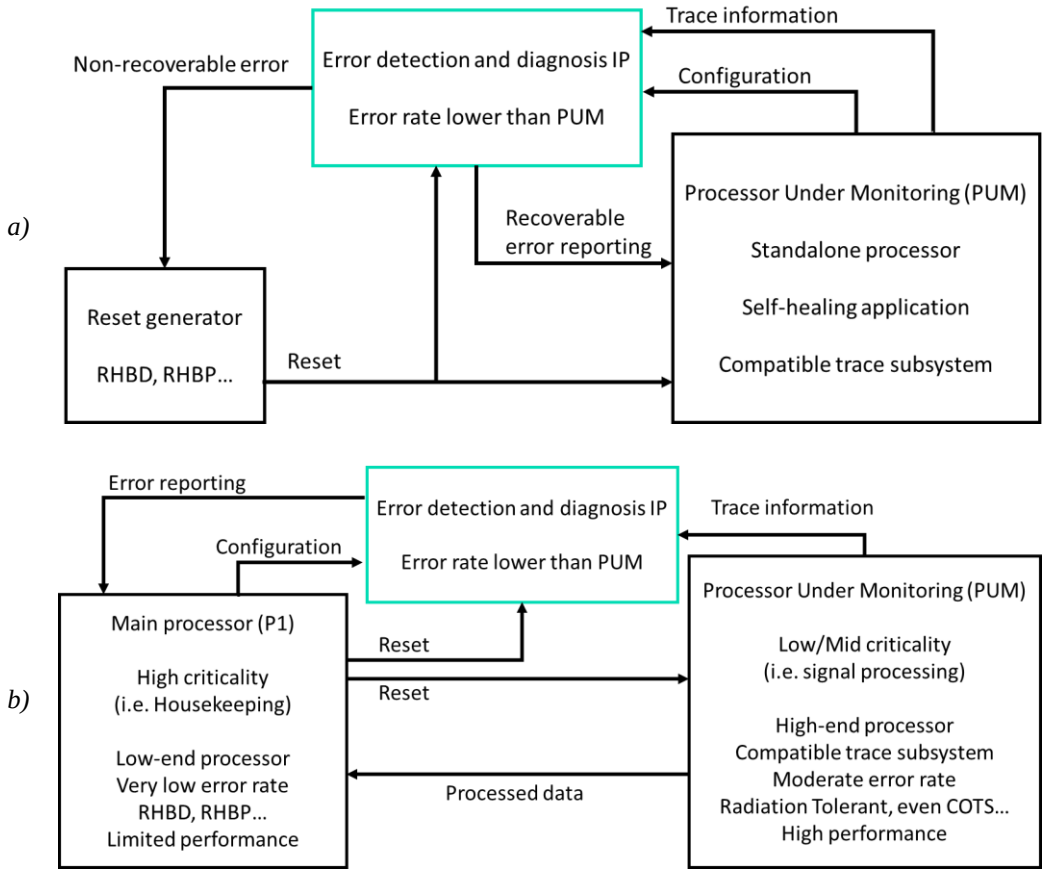

Figure 3. IP integrated in a) binary and b) ternary architecture configuration

reset to avoid a permanent functional interrupt. A binary architecture is the minimum fault tolerant system that can be built around this IP and requires effort from the designer to assess that the whole system would meet the dependability requirements. Binary system architecture is depicted in Fig. 3 a).

In the case of a **ternary architecture** configuration, the IP is checking the execution of a processor, PUM, which is only in charge of performing heavy, non-critical tasks which require very high performance. An additional microprocessor, P1, is governing the entire system without supervision, so it must be expected to have very low error rate and provide all safety and time critical tasks to meet dependability requirements. However, there is probably no need for P1 to be extremely powerful because it can rely on PUM to perform all heavy, non-critical tasks. The IP will inform P1 whenever and error is found on PUM to take a corrective action. In this case, the designer effort is lower as the corrective action can be as simple as ignoring the last data packet or even a PUM reset, since PUM is not servicing any critical task. Ternary system architecture is depicted in Fig. 3 b).

Several works have been conducted by the authors following the described trace monitoring approach using Xilinx Zynq-7000 AP SoC during the development of the IP. [21] and [22] demonstrated the feasibility of using trace information for error detection purposes in both control-flow and data with several application benchmarks, reaching up to 95% error coverage. Later works demonstrated the capability of the IP to be integrated in a more realistic application and to be combined with other hardening techniques such as dual core lockstep [23] and data redundancy acceleration using SIMD [24], achieving up to 99.9% error coverage. Most recent works illustrate the error diagnosis capabilities of the IP under proton and neutron irradiation [25] and also under laser fault injection [26], demonstrating fine granularity on discriminating error types, and the suitability of the recorded information to perform effective error diagnosis.

5. CONCLUSIONS

Increasingly competitive space industry constantly seeks for new solutions to enhance spacecraft processing capabilities on orbit. COTS processors, and particularly ARM cores, are receiving much attention in the last years due to their excellent performance and power consumption features. Despite COTS processors have been flown on successful missions, several challenges still prevent COTS processors to be massively adopted in space missions, as they involve risks regarding radiation hardness assurance.

Providing solutions to ease the safe introduction of COTS processors on spacecraft may enable unprecedented computing capabilities on orbit, leading to a more efficient use of resources. This paper has presented a new error detection and diagnosis technique based on trace information monitoring and an IP design to implement it.

Trace monitoring is a new tool in the designer's toolbox to manage risks and improve the reliability of microprocessor-based space systems. This solution is currently available at ARQUIMEA as an IP core compatible with ARM Cortex-A9 processor. It has been functionally validated in Xilinx Zynq device under radiation testing (TRL3-4) obtaining high error detection rate (up to 99.9%) [24] and useful diagnosis information [25][26]. The IP features low pin count and parametric design ready to be implemented in any FPGA with low footprint. Currently, efforts are ongoing to enhance IP capabilities and compatibility with a wider range of technologies and processor cores, including Xilinx Zynq Ultrascale, Microchip rad-tolerant devices and NanoXplore FPGAs.

ACKNOWLEDGEMENTS

This work has been supported in part by the Spanish Ministry of Science and Innovation under project PID2019-106455GB-C21 and by the Community of Madrid under grant IND2017/TIC-7776. The IP has been developed in collaboration between Universidad Carlos III de Madrid and Arquimea, in the framework of an Industrial Ph.D. program.

REFERENCES

[1] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", IEEE Trans. on Dev. and Materials Rel., vol. 5, no. 3, pp. 305-316, 2005.

[2] R. Ginosar, "Survey of processors for space", Proc. Int. Space System Engineering Conf. (DASIA), 1B, 2012.

[3] K. A. LaBel *et al.*, "Emerging Radiation Hardness Assurance Issues: A NASA Approach for Spaceflight Programs", IEEE Trans. Nucl. Sci. vol. 45, 2727(1998).

[4] K. A. LaBel, "NEPP Roadmaps, COTS, and Small Missions", Presented at NEPP Electronics Technology Workshop (ETW), Jun., 2016.

[5] M. Nicolaidis, "Soft errors in modern electronic systems", Springer, 2011.

[6] J. R. Azambuja, *et al.*, "Exploring the limitations of software-only techniques in SEE detection coverage", Journal of Electronic Testing, no. 27, pp. 541–550, 2011.

[7] H. Quinn, "Challenges in testing complex systems", IEEE Trans. Nucl. Sci., vol. 61, no. 2, pp. 766-786, Apr. 2014.

[8] P. Cheynet, *et al.*, "Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors", IEEE Trans. Nucl. Sci., vol. 47, no. 6, pp. 2231–2236, Dic. 2000.

[9] E. Chielle, *et al.*, "Evaluating Selective Redundancy in Data-Flow Software-Based Techniques", IEEE Trans. Nucl. Sci., vol. 60, no. 4, pp. 2768-2775, Aug. 2013.

[10] M. Hiller, "Executable assertions for detecting data errors in embedded control systems", Proceedings of the IEEE Intl. Conf. on Dependable Systems and Networks, pp 24–33, 2000.

[11] A. Benso, *et al.*, "A C/C++ source-to-source compiler for dependable applications", IEEE Intl. Conf. on Dependable Systems and Networks, pp. 71-78, 2000.

[12] M. Pignol, "DMT and DT2: Two Fault-Tolerant Architectures developed by CNES for COTS-based Spacecraft Supercomputers", Proc. 12th Int. On-Line Testing Symp. (IOLTS), pp. 203-212, 2006.

[13] A. B. de Oliveira *et al.*, "Lockstep Dual-Core ARM A9: Implementation and Resilience Analysis Under Heavy Ion-Induced Soft Errors", IEEE Trans. Nucl. Sci., vol. 65, no. 8, pp. 1783-1790, Aug. 2018.

[14] S. S. Mukherjee, *et al.*, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor", Proc. 36th Annual IEEE/ACM Intl. Symp. on Microarchitecture (MICRO-36), pp. 29-40, Dec. 2003.

[15] J. M. Mogollon, *et al.*, "Real Time SEU Detection and Diagnosis for Safety or Mission-Critical ICs Using HASH Library-Based Fault Dictionaries". Proc. RADECS, paper J-3, pp.705-710, Sept. 2011.

[16] M. Grosso, *et al.*, "An on-line fault detection technique based on embedded debug features", Proc.16th IEEE On-Line Testing Symp., pp. 167-172, 2010.

[17] A. Lindoso, *et al.*, "A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA", IEEE Trans. Nucl. Sci., vol. 64, no. 1, pp. 374-381, Jan. 2017.

[18] L. Entrena, *et al.*, "Fault-tolerance techniques for soft-core processors using the Trace Interface". In "FPGAs and Parallel Architectures for Aerospace Applications. Soft Errors and Fault-Tolerant Design", Springer, 2016.

[19] "CoreSight Components. Technical Reference Manual", ARM Ltd., DDI 0314H, 2009.

[20] "Zynq-7000 All Programmable SoC: Technical Reference Manual", Xilinx Inc., Technical Reference Manual UG585, Sept. 2016.

[21] M. Peña-Fernandez, *et al.*, "PTM-based hybrid error-detection architecture for ARM microprocessors", Microelectronics Reliability, 88, pp. 925-930, 2018.

[22] M. Peña-Fernandez, *et al.*, "Online error detection through trace infrastructure in ARM microprocessors", IEEE Trans. Nucl. Sci., vol. 66, no. 7, pp. 1457-1464, July 2019.

[23] M. Peña-Fernández, *et al.*, "Dual-Core Lockstep enhanced with redundant multithread support and control-flow error detection", Microelectronics Reliability, vol. 100–101, Article No. 113447, Sept. 2019.

[24] M. Peña-Fernandez, *et al.*, "Error Detection and Mitigation of Data-Intensive Microprocessor Applications Using SIMD and Trace Monitoring", IEEE Trans. on Nucl. Sci., vol. 67, no. 7, pp. 1452 - 1460, Jul. 2020.

[25] M. Peña-Fernandez, *et al.*, "The Use of Microprocessor Trace Infrastructures for Radiation-Induced Fault Diagnosis", IEEE Trans. on Nucl. Sci., vol. 67, no. 1, pp. 126-134, Jan. 2020.

[26] M. Peña-Fernandez, *et al.*, "Microprocessor Error Diagnosis by Trace Monitoring under Laser Testing", IEEE Trans. on Nucl. Sci., (Early Access).