Error Detection and Diagnosis for System-on-Chip in Space Applications

by

Manuel Peña Fernández

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Electrical Engineering, Electronics and Automation

Universidad Carlos III de Madrid

Advisors:

Dra. Almudena Lindoso Muñoz
Dr. Luis Alfonso Entrena Arrontes

Tutor:

Dr. Luis Alfonso Entrena Arrontes

June 2022

This Thesis is distributed under license “Creative Commons **Attribution - Non Commercial - Non Derivatives**”.

A mi familia

AGRADECIMIENTOS

En primer lugar, quisiera aprovechar estas líneas para expresar mi agradecimiento a todos aquellos que me han acompañado y ayudado durante el desarrollo y escritura de esta Tesis. Gracias especialmente a mis directores, Almudena y Luis, por su gran dedicación en este trabajo. Vuestra ayuda, orientación y experiencia han enriquecido los resultados obtenidos y me han empujado a dar de mí lo mejor.

Gracias a todos mis compañeros de universidad, por su compañerismo y amistad durante toda esta etapa. En especial a mis amigos Jorge Plaza, Dragos Andrei Poiana, Jorge Rodríguez, Alejandro Ruiz y Carlos Romero. Sé que compartís mi entusiasmo por haber finalizado este trabajo.

Gracias a todos los profesores que he tenido, por sembrar en mí la inquietud por lo desconocido y demostrarme la satisfacción que se puede obtener del trabajo bien hecho.

Gracias a la Universidad Carlos III de Madrid, en cuyo seno he realizado mi formación superior, incluida esta Tesis.

Mi sincero agradecimiento a todas las personas de Arquimea, que me aceptaron para desarrollar esta investigación como un Doctorado Industrial dentro de una prestigiosa empresa del ámbito espacial. Estoy muy agradecido a todos los compañeros con los que he tenido el placer de trabajar, por su profesionalidad y amabilidad. Estoy seguro de que he aprendido muchas cosas de cada uno de vosotros. Gracias en especial a Daniel González, como mi tutor dentro de la empresa, y a Francisco Álvarez y a José Ángel Domínguez por su continuo apoyo en este trabajo durante estos años.

Además, me gustaría agradecer a la Comunidad de Madrid por financiar públicamente la investigación doctoral. Este trabajo ha sido financiado en parte por la Comunidad de Madrid mediante una beca, con número de expediente IND2017/TIC-7776, concedida en 2017 dentro de una convocatoria de ayudas para la realización de doctorados industriales.

A todos mis amigos, por seguir ahí, entusiasmándonos aún más con nuestros logros. Formáis parte también de esta historia.

A mi familia, por creer siempre en mí y apoyarme incondicionalmente durante esta etapa.

A Patricia, por quererme tantísimo.

PUBLISHED AND SUBMITTED CONTENT

1st-author journal articles

[J1] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, S. Philippe, Y. Morilla, and P. Martin-Holgado, “PTM-based hybrid error-detection architecture for ARM microprocessors”, *Microelectronics Reliability*, vol. 88-90, pp. 925–930, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.074 (JCR Q3). This article has been wholly included in this Thesis in Chapter 4.

[J2] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martín-Holgado, “Online error detection through trace infrastructure in ARM microprocessors”, *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1457–1464, Jul. 2019. DOI: 10.1109/TNS.2019.2921767 (JCR Q2). This article has been wholly included in this Thesis in Chapter 5.

[J3] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, M. García-Valderas, L. Entrena, A. Martínez-Álvarez, and S. Cuenca-Asensi, “Dual-core lockstep enhanced with redundant multithread support and control-flow error detection”, *Microelectronics Reliability*, vol. 100-101, no. 113447, Sep. 2019. DOI: 10.1016/j.microrel.2019.113447 (JCR Q3). This article has been wholly included in this Thesis in Chapter 6.

[J4] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, “The use of microprocessor trace infrastructures for radiation-induced fault diagnosis”, *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 126–134, Jan. 2020. DOI: 10.1109/TNS.2019.2956204 (JCR Q2). This article has been wholly included in this Thesis in Chapter 7.

[J5] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, “Error detection and mitigation of data-intensive microprocessor applications using SIMD and trace monitoring”, *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1452–1460, Jul. 2020. DOI: 10.1109/TNS.2020.2992299 (JCR Q2). This article has been wholly included in this Thesis in Chapter 8.

[J6] M. Peña-Fernandez, A. Lindoso, L. Entrena, I. Lopes, and V. Pouget, “Microprocessor error diagnosis by trace monitoring under laser testing”, *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1651–1659, Aug. 2021. DOI: 10.1109/TNS.2021.3067554 (JCR Q2). This article has been wholly included in this Thesis in Chapter 9.

[J7] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, S. Cuenca-Asensi, L. Entrena, Y. Morilla, P. Martín-Holgado, and A. Martínez-Álvarez, "Hybrid lockstep technique for soft error mitigation", *IEEE Transactions on Nuclear Science*, 2022. DOI: 10.1109/TNS.2022.3149867 (JCR Q2). This article has been partly included in this Thesis in Chapter 3.

1st-author conference articles

[C1] M. Peña-Fernandez, A. Lindoso, and L. Entrena, "IP to detect and diagnose errors in COTS microprocessors through the Trace Interface", *presented at the 2nd European Workshop on On-Board Data Processing (OBDP2021)*, Jun. 2021. DOI: 10.5281/zenodo.5521538. This article has been wholly included in this Thesis in Chapter 10.

Whenever material from any of these sources is included in this Thesis, it is singled out with typographic means and an explicit reference.

OTHER RESEARCH MERITS

Participation in specialized forums

- Poster communication at 29th European Conference on Radiation and its Effects on Components and Systems (RADECS), held in Gothenburg, Sweden on September 16-21, 2018.
- Attendance and oral presentation at 29th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF), held in Aalborg, Denmark on October 1-5, 2018.
- Attendance to 14th international School on the Effects of Radiation on Embedded Systems for Space Applications (SERESSA), held at ESA ESTEC Campus in Noordwijk, The Netherlands on November 12-16, 2018.
- Poster communication at 56th IEEE Nuclear and Space Radiation Effects Conference (NSREC), held in San Antonio, Texas, United States on July 8-12, 2018.
- Attendance to RADECS 2019 Short Course, held in Montpellier, France on September 16th, 2019.
- Attendance and oral presentation at 30th European Conference on Radiation and its Effects on Components and Systems (RADECS), held in Montpellier, France on September 16-20, 2019.
- Attendance and oral presentation at 30th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF), held in Toulouse, France on September 23-26, 2019.
- Attendance and oral presentation at 31st European Conference on Radiation and its Effects on Components and Systems (RADECS), held online on October 19th - November 20th, 2020.
- Attendance to NSREC 2020 Short Course, held online on November 29-30, 2020.
- Attendance to the 57th IEEE Nuclear and Space Radiation Effects Conference (NSREC), held online on December 01-30, 2020.
- Attendance and oral presentation at the 2nd European Workshop on On-Board Data Processing OBDP 2021 held online on June 14-17, 2021.
- Attendance and oral presentation at the 32nd European Conference on Radiation and its Effects on Components and Systems (RADECS), held in Vienna, Austria on September 13-17, 2021.

RESUMEN NO TÉCNICO

Los componentes electrónicos comerciales, comúnmente llamados componentes *Commercial-Off-The-Shelf* (COTS) están presentes en multitud de dispositivos habituales en nuestro día a día. Particularmente, el uso de microprocesadores y sistemas en chip (SoC) altamente integrados ha favorecido la aparición de dispositivos electrónicos cada vez más inteligentes que sostienen el estilo de vida y el avance de la sociedad moderna. Su uso se ha generalizado incluso en aquellos sistemas que se consideran críticos para la seguridad, como vehículos, aviones, armamento, dispositivos médicos, implantes o centrales eléctricas. En cualquiera de ellos, un fallo podría tener graves consecuencias humanas o económicas. Sin embargo, todos los sistemas electrónicos conviven constantemente con factores internos y externos que pueden provocar fallos en su funcionamiento. La capacidad de un sistema para funcionar correctamente en presencia de fallos se denomina tolerancia a fallos, y es un requisito en el diseño y operación de sistemas críticos.

Los vehículos espaciales como satélites o naves espaciales también hacen uso de microprocesadores para operar de forma autónoma o semi autónoma durante su vida útil, con la dificultad añadida de que no pueden ser reparados en órbita, por lo que se consideran sistemas críticos. Además, las duras condiciones existentes en el espacio, y en particular los efectos de la radiación, suponen un gran desafío para el correcto funcionamiento de los dispositivos electrónicos. Concretamente, los fallos transitorios provocados por radiación (conocidos como *soft errors*) tienen el potencial de ser una de las mayores amenazas para la fiabilidad de un sistema en el espacio.

Las misiones espaciales de gran envergadura, típicamente financiadas públicamente como en el caso de la NASA o la Agencia Espacial Europea (ESA), han tenido históricamente como requisito evitar el riesgo a toda costa por encima de cualquier restricción de coste o plazo. Por ello, la selección de componentes resistentes a la radiación (*rad-hard*) específicamente diseñados para su uso en el espacio ha sido la metodología imperante en el paradigma que hoy podemos denominar industria espacial tradicional, u *Old Space*. Sin embargo, los componentes *rad-hard* tienen habitualmente un coste mucho más alto y unas prestaciones mucho menores que otros componentes COTS equivalentes. De hecho, los componentes COTS ya han sido utilizados satisfactoriamente en misiones de la NASA o la ESA cuando las prestaciones requeridas por la misión no podían ser cubiertas por ningún componente *rad-hard* existente.

En los últimos años, el acceso al espacio se está facilitando debido en gran parte a la entrada de empresas privadas en la industria espacial. Estas empresas no siempre buscan evitar el riesgo a toda costa, sino que deben perseguir una rentabilidad económica, por lo que hacen un balance entre riesgo, coste y plazo mediante gestión del riesgo en un paradigma denominado Nuevo Espacio o *New Space*. Estas empresas a menudo están interesadas en entregar servicios basados en el espacio con las máximas prestaciones y el

mayor beneficio posibles, para lo cual los componentes *rad-hard* son menos atractivos debido a su mayor coste y menores prestaciones que los componentes COTS existentes.

Sin embargo, los componentes COTS no han sido específicamente diseñados para su uso en el espacio y típicamente no incluyen técnicas específicas para evitar que los efectos de la radiación afecten su funcionamiento. Los componentes COTS se comercializan tal cual son, y habitualmente no es posible modificarlos para mejorar su resistencia a la radiación. Además, los elevados niveles de integración de los sistemas en chip (SoC) complejos de altas prestaciones dificultan su observación y la aplicación de técnicas de tolerancia a fallos. Este problema es especialmente relevante en el caso de los microprocesadores. Por tanto, existe un gran interés en el desarrollo de técnicas que permitan conocer y mejorar el comportamiento de los microprocesadores COTS bajo radiación sin modificar su arquitectura y sin interferir en su funcionamiento para facilitar su uso en el espacio y con ello maximizar las prestaciones de las misiones espaciales presentes y futuras.

En esta Tesis se han desarrollado técnicas novedosas para detectar, diagnosticar y mitigar los errores producidos por radiación en microprocesadores y sistemas en chip (SoC) comerciales, utilizando la interfaz de traza como punto de observación. La interfaz de traza es un recurso habitual en los microprocesadores modernos, principalmente enfocado a soportar las tareas de desarrollo y depuración del software durante la fase de diseño. Sin embargo, una vez el desarrollo ha concluido, la interfaz de traza típicamente no se utiliza durante la fase operativa del sistema, por lo que puede ser reutilizada sin coste. La interfaz de traza constituye un punto de conexión viable para observar el comportamiento de un microprocesador de forma no intrusiva y sin interferir en su funcionamiento.

Como resultado de esta Tesis se ha desarrollado un módulo IP capaz de recabar y decodificar la información de traza de un microprocesador COTS moderno de altas prestaciones. El IP es altamente configurable y personalizable para adaptarse a diferentes aplicaciones y tipos de procesadores. Ha sido diseñado y validado utilizando el dispositivo Zynq-7000 de Xilinx como plataforma de desarrollo, que constituye un dispositivo COTS de interés en la industria espacial. Este dispositivo incluye un procesador ARM Cortex-A9 de doble núcleo, que es representativo del conjunto de microprocesadores *hard-core* modernos de altas prestaciones. El IP resultante es compatible con la tecnología ARM CoreSight, que proporciona acceso a información de traza en los microprocesadores ARM. El IP incorpora técnicas para detectar errores en el flujo de ejecución y en los datos de la aplicación ejecutada utilizando la información de traza, en tiempo real y con muy baja latencia. El IP se ha validado en campañas de inyección de fallos y también en radiación con protones y neutrones en instalaciones especializadas. También se ha combinado con otras técnicas de tolerancia a fallos para construir técnicas híbridas de mitigación de errores. Los resultados experimentales obtenidos demuestran su alta capacidad de detección y potencialidad en el diagnóstico de errores producidos por radiación.

El resultado de esta Tesis, desarrollada en el marco de un Doctorado Industrial entre la Universidad Carlos III de Madrid (UC3M) y la empresa Arquimea, se ha transferido

satisfactoriamente al entorno empresarial en forma de un proyecto financiado por la Agencia Espacial Europea para continuar su desarrollo y posterior explotación.

Palabras clave: Fiabilidad, Tolerancia a fallos, Fallos transitorios, Radiación, Detección de errores, Diagnóstico de errores, Mitigación de errores, SoC, COTS, ARM, Microprocesador, CoreSight, Interfaz de traza.

NON-TECHNICAL SUMMARY

Commercial electronic components, also known as Commercial-Off-The-Shelf (COTS), are present in a wide variety of devices commonly used in our daily life. Particularly, the use of microprocessors and highly integrated System-on-Chip (SoC) devices has fostered the advent of increasingly intelligent electronic devices which sustain the lifestyles and the progress of modern society. Microprocessors are present even in safety-critical systems, such as vehicles, planes, weapons, medical devices, implants, or power plants. In any of these cases, a fault could involve severe human or economic consequences. However, every electronic system deals continuously with internal and external factors that could provoke faults in its operation. The capacity of a system to operate correctly in presence of faults is known as fault-tolerance, and it becomes a requirement in the design and operation of critical systems.

Space vehicles such as satellites or spacecraft also incorporate microprocessors to operate autonomously or semi-autonomously during their service life, with the additional difficulty that they cannot be repaired once in-orbit, so they are considered critical systems. In addition, the harsh conditions in space, and specifically radiation effects, involve a big challenge for the correct operation of electronic devices. In particular, radiation-induced soft errors have the potential to become one of the major risks for the reliability of systems in space.

Large space missions, typically publicly funded as in the case of NASA or European Space Agency (ESA), have followed historically the requirement to avoid the risk at any expense, regardless of any cost or schedule restriction. Because of that, the selection of radiation-resistant components (known as rad-hard) specifically designed to be used in space has been the dominant methodology in the paradigm of traditional space industry, also known as “Old Space”. However, rad-hard components have commonly a much higher associated cost and much lower performance that other equivalent COTS devices. In fact, COTS components have already been used successfully by NASA and ESA in missions that requested such high performance that could not be satisfied by any available rad-hard component.

In the recent years, the access to space is being facilitated in part due to the irruption of private companies in the space industry. Such companies do not always seek to avoid the risk at any cost, but they must pursue profitability, so they perform a trade-off between risk, cost, and schedule through risk management in a paradigm known as “New Space”. Private companies are often interested in deliver space-based services with the maximum performance and maximum benefit as possible. With such objective, rad-hard components are less attractive than COTS due to their higher cost and lower performance.

However, COTS components have not been specifically designed to be used in space and typically they do not include specific techniques to avoid or mitigate the radiation

effects in their operation. COTS components are commercialized "as is", so it is not possible to modify them to improve their susceptibility to radiation effects. Moreover, the high levels of integration of complex, high-performance SoC devices hinder their observability and the application of fault-tolerance techniques. This problem is especially relevant in the case of microprocessors. Thus, there is a growing interest in the development of techniques allowing to understand and improve the behavior of COTS microprocessors under radiation without modifying their architecture and without interfering with their operation. Such techniques may facilitate the use of COTS components in space and maximize the performance of present and future space missions.

In this Thesis, novel techniques have been developed to detect, diagnose, and mitigate radiation-induced errors in COTS microprocessors and SoCs using the trace interface as an observation point. The trace interface is a resource commonly found in modern microprocessors, mainly intended to support software development and debugging activities during the design phase. However, it is commonly left unused during the operational phase of the system, so it can be reused with no cost. The trace interface constitutes a feasible connection point to observe microprocessor behavior in a non-intrusive manner and without disturbing processor operation.

As a result of this Thesis, an IP module has been developed capable to gather and decode the trace information of a modern, high-end, COTS microprocessor. The IP is highly configurable and customizable to support different applications and processor types. The IP has been designed and validated using the Xilinx Zynq-7000 device as a development platform, which is an interesting COTS device for the space industry. This device features a dual-core ARM Cortex-A9 processor, which is a good representative of modern, high-end, hard-core microprocessors. The resulting IP is compatible with the ARM CoreSight technology, which enables access to trace information in ARM microprocessors. The IP is able to detect errors in the execution flow of the microprocessor and in the application data using trace information, in real time and with very low latency. The IP has been validated in fault injection campaigns and also under proton and neutron irradiation campaigns in specialized facilities. It has also been combined with other fault-tolerance techniques to build hybrid error mitigation approaches. Experimental results demonstrate its high detection capabilities and high potential for the diagnosis of radiation-induced errors.

The result of this Thesis, developed in the framework of an Industrial Ph.D. between the University Carlos III of Madrid (UC3M) and the company Arquimea, has been successfully transferred to the company business as a project sponsored by European Space Agency to continue its development and subsequent commercialization.

Keywords: Reliability, Fault-tolerance, Soft errors, Radiation, Error detection, Error diagnosis, Error mitigation, SoC, COTS, ARM, Microprocessor, CoreSight, Trace interface.

RESUMEN TÉCNICO

La miniaturización de las tecnologías de fabricación de circuitos integrados y el asociado aumento de sus prestaciones han favorecido en los últimos años un desarrollo sin precedentes de los sistemas electrónicos. Los componentes electrónicos comerciales, comúnmente llamados componentes *Commercial-Off-The-Shelf* (COTS) están presentes en multitud de dispositivos habituales en nuestro día a día. Particularmente, el uso de microprocesadores y sistemas en chip (SoC) altamente integrados ha favorecido la aparición de dispositivos electrónicos cada vez más inteligentes que aceleran el avance de nuestra sociedad. Su uso se ha generalizado incluso en aquellos sistemas que se consideran críticos para la seguridad, como vehículos, aviones, armamento, dispositivos médicos, implantes o centrales eléctricas. En cualquiera de ellos, un fallo podría tener graves consecuencias humanas o económicas, por lo que la tolerancia a fallos se convierte en un requisito para su diseño y operación.

La tolerancia a fallos es una disciplina especialmente relevante en la industria espacial, ya que los vehículos y naves espaciales generalmente no pueden ser reparados en órbita y hacen uso extensivo de sistemas electrónicos para su funcionamiento. En el espacio, la radiación afecta a los componentes electrónicos mediante mecanismos acumulativos como la dosis ionizante total (*Total Ionizing Dose*, TID) o la dosis de deterioro por desplazamiento (*Displacement Damage Dose*, DDD), que deterioran los parámetros funcionales de los circuitos integrados progresivamente hasta que finalmente dejan de funcionar. Sin embargo, la radiación también produce efectos aleatorios conocidos como efectos de evento único (*Single-Event Effects*, SEE), por los cuales una única partícula incidente en el dispositivo puede producir fallos permanentes (conocidos como *hard errors*), o transitorios (conocidos como *soft errors*). Los efectos que produce la radiación en los componentes electrónicos se han estudiado detalladamente desde la carrera espacial y se han desarrollado técnicas de endurecimiento cada vez más eficientes para evitar, reducir o mitigar su impacto en el funcionamiento de los sistemas espaciales.

Durante la segunda mitad del siglo XX, la industria espacial estaba formada mayoritariamente por grandes agencias gubernamentales, como la NASA o la Agencia Espacial Europea (ESA), que desarrollaban programas espaciales de gran envergadura que tenían como objetivo evitar el riesgo a toda costa, por encima de cualquier consideración de plazo o coste. Esta metodología pretendía evitar el fracaso de la misión, que podría comprometer no solo vidas humanas, sino también inversiones millonarias en el caso de misiones no tripuladas y afectar a la imagen y el apoyo social de los programas espaciales. Por ello, seleccionaban preferentemente componentes resistentes a la radiación, conocidos como *rad-hard*, específicamente diseñados para su uso en el espacio mediante técnicas de endurecimiento y capaces de funcionar correctamente bajo condiciones de radiación. Sin embargo, debido a su pequeño nicho de mercado y la dificultad añadida

en su desarrollo, los componentes *rad-hard* tienen habitualmente un coste mucho más alto y ofrecen prestaciones mucho menores que sus equivalentes COTS. De hecho, los componentes COTS ya han sido utilizados satisfactoriamente en misiones de la NASA o la ESA cuando las prestaciones requeridas por la misión no podían ser cubiertas por ningún componente *rad-hard* existente.

En los últimos años se está viviendo un cambio de paradigma en la industria espacial, propiciado por la entrada masiva de empresas privadas que está facilitando el acceso al espacio. Estas empresas persiguen generalmente objetivos de rentabilidad y buscan maximizar el beneficio como proveedores de servicios basados en el espacio. En este contexto, se busca un balance entre riesgo, coste y plazo mediante la gestión del riesgo, en una tendencia que se conoce como Nuevo Espacio o *New Space*. Los componentes COTS resultan más atractivos para las empresas del *New Space*, que buscan diferenciarse en un mercado cada vez más competitivo ofreciendo las prestaciones más avanzadas posibles y la reducción de los costes de operación. Estos objetivos en la mayoría de las ocasiones no son alcanzables mediante el uso exclusivo de componentes *rad-hard*, por lo que existe un interés creciente por el uso de COTS en el espacio.

Sin embargo, los componentes COTS no han sido diseñados teniendo en cuenta su posible uso en el espacio, por lo que típicamente apenas integran recursos o técnicas para hacer frente a los efectos de la radiación. Además, factores relacionados con la miniaturización de las tecnologías del silicio como el incremento de la densidad de integración, la reducción de las tensiones de alimentación, y el estrechamiento de los márgenes de ruido, así como el aumento en la frecuencia de funcionamiento de los dispositivos electrónicos, los ha hecho cada vez más vulnerables a los fallos transitorios producidos por radiación (*soft errors*). Estos fallos, que en el pasado estaban principalmente asociados con la industria espacial, son actualmente un desafío para los sistemas electrónicos que desempeñan funciones críticas para la seguridad en aviónica o incluso en aplicaciones en tierra. Se considera que los fallos transitorios producidos por radiación tienen el potencial de ser una de las mayores amenazas para la fiabilidad de un sistema en el espacio y su mitigación es de gran interés para permitir un mayor uso de componentes COTS en futuras misiones.

Existen numerosas técnicas de endurecimiento para proteger los dispositivos electrónicos contra la radiación, o mitigar sus efectos. La gran mayoría de estas técnicas utilizan esquemas redundantes que consisten en repetir la misma estructura u operación en múltiples instancias y comparar los resultados proporcionados por cada réplica. Los dispositivos *rad-hard* típicamente utilizan redundancia hardware para enmascarar o mitigar fallos transitorios de forma transparente al usuario. Sin embargo, no es posible aplicar esta metodología en componentes COTS, ya que se comercializan tal cual son y su arquitectura no puede ser modificada para mejorar su resistencia a los efectos de la radiación. Además, los detalles de su arquitectura interna son en muchas ocasiones desconocidos, ya que los fabricantes los mantienen en secreto para proteger su ventaja competitiva. Por tanto, existe un gran interés en el desarrollo de técnicas que permitan conocer y mejorar el

comportamiento de los componentes COTS bajo radiación sin modificar su arquitectura y sin interferir en su funcionamiento para facilitar su uso en el espacio y con ello maximizar las prestaciones de las misiones espaciales presentes y futuras.

El desarrollo de técnicas de tolerancia a fallos para microprocesadores COTS es un desafío. Históricamente se han utilizado los buses de memoria como punto de observación, pero esto puede no resultar viable sobre todo en el caso de microprocesadores *hard-core* implementados en sistemas monolíticos altamente integrados como es el caso de los sistemas en chip (*System-on-Chip*, SoC). Habitualmente, este tipo de dispositivos presentan una cantidad limitada de interfaces disponibles para observar el comportamiento del procesador. La observabilidad es una característica fundamental para conocer el comportamiento de cualquier dispositivo de cara a poder detectar comportamientos anómalos y eventualmente ser capaz de aplicar técnicas de tolerancia a fallos de forma eficaz. La observabilidad de un microprocesador disminuye generalmente conforme aumentan su complejidad así como su nivel de integración monolítica en sistemas SoC.

En esta Tesis se han desarrollado técnicas novedosas para detectar, diagnosticar y mitigar los errores producidos por radiación en microprocesadores y sistemas en chip (SoC) comerciales, con el objetivo de mejorar su tolerancia a fallos. Como principal novedad, las técnicas desarrolladas utilizan la interfaz de traza como punto de observación, obteniendo una alta compatibilidad con sistemas SoC altamente integrados. La interfaz de traza es recurso habitual en los microprocesadores modernos, principalmente enfocado a soportar las tareas de desarrollo y depuración del software durante la fase de diseño, realizando tareas como pruebas de planificabilidad o cobertura de código. Sin embargo, una vez el desarrollo software ha concluido, la interfaz de traza típicamente no se utiliza durante la fase operativa del sistema, por lo que puede ser reutilizada sin coste. La interfaz de traza constituye un punto de conexión viable para observar el comportamiento de un microprocesador de forma no intrusiva y sin interferir en su funcionamiento.

Como resultado de esta Tesis se ha desarrollado un módulo IP capaz de recabar y decodificar la información de traza de un microprocesador COTS moderno de altas prestaciones. El IP es altamente configurable y personalizable para adaptarse a diferentes aplicaciones y tipos de procesadores. Ha sido diseñado en lenguaje VHDL y validado utilizando el dispositivo SoC Zynq-7000 de Xilinx como plataforma de desarrollo, que constituye un dispositivo COTS de interés en la industria espacial. Este dispositivo incluye un procesador de doble núcleo ARM Cortex-A9, que es representativo del conjunto de microprocesadores hard-core modernos de altas prestaciones. El IP resultante es compatible con la tecnología ARM CoreSight, que proporciona acceso a información de traza en los microprocesadores ARM. El IP desarrollado es capaz de decodificar la información de traza producida por la macrocelda de traza de programa (*Program Trace Macrocell*, PTM) y la macrocelda de traza de instrumentación (*Instrumentation Trace Macrocell*, ITM), ambas presentes en el dispositivo Zynq-7000. Utilizando la información obtenida mediante el puerto de traza (*Trace Port Interface Unit*, TPIU), el IP es capaz de detectar errores en el flujo de ejecución del procesador, incluyendo ambos núcleos presentes en el dispositivo

Zynq-7000, así como en los datos de la aplicación ejecutada, en tiempo real y con muy baja latencia. El IP incluye todas las técnicas de detección y diagnóstico de errores diseñadas en esta Tesis, que se han desarrollado, probado, evaluado e incorporado progresivamente:

- Técnicas de detección de errores de control de flujo de ejecución:
  - Comprobación de rango de direcciones: El IP es capaz de obtener los valores sucesivos del contador de programa (PC) a partir de la información de traza para evaluar en todo momento si el microprocesador se encuentra ejecutando código dentro de una zona de memoria previamente definida como válida. Esta zona es configurable por el usuario mediante registros de configuración en el IP, que delimitan los rangos de direcciones válidas. En caso de que el IP detecte que el microprocesador ejecuta alguna instrucción fuera de la zona permitida, este es capaz de detectarlo con muy baja latencia e indicarlo mediante una señal de error. El IP puede realizar esta comprobación por cada uno de los microprocesadores o núcleos existentes en el dispositivo de forma independiente.
  - Comprobación temporal de direcciones: Utilizando la información del contador de programa (PC), el IP es capaz de determinar si el procesador está ejecutando la aplicación correctamente desde un punto de vista temporal. Mediante registros configurables, el usuario indica un valor de PC cíclico dentro del programa, que puede estar asociado con el bucle principal del mismo. Si el IP detecta que ha pasado un tiempo excesivo, configurable, desde la última vez que el microprocesador hizo un ciclo completo del bucle principal, este lo indica con una señal de error. Esta técnica es muy útil para detectar con baja latencia situaciones en las que el microprocesador ha perdido el control como consecuencia de un error o una excepción no prevista. El IP puede realizar esta comprobación por cada uno de los microprocesadores o núcleos existentes en el dispositivo de forma independiente.
- Técnicas de detección de errores de datos:
  - Comprobación de rango de datos: El IP es capaz de obtener el valor de las variables del programa a partir de la información de traza para evaluar en cualquier momento su validez dentro de la aplicación. El usuario puede configurar distintos rangos de valores como válidos para ciertas variables utilizando registros de configuración. El IP comprueba automáticamente si el valor de las variables deseadas se encuentra dentro del rango permitido, y en caso negativo lo indica mediante una señal de error.
  - Comprobación de consistencia de datos: Utilizando la información del valor de las variables del programa, el IP puede hacer comprobaciones de consistencia para datos duplicados o triplicados, contribuyendo a una mejora de prestaciones del sistema ya que descarga al microprocesador de realizar esta tarea. El IP

puede comprobar automáticamente relaciones de comparación entre grupos de dos y tres valores de datos para indicar mediante una señal de error el momento en el que cualquiera de las relaciones no se cumple.

- Técnicas de diagnóstico de errores:
  - La riqueza de la información de traza trasciende el mero propósito de detección de errores ya que, una vez detectados, permite contextualizarlos con el resto del funcionamiento del dispositivo simplemente comprobando la información de traza inmediatamente anterior al error. Mediante el análisis de esta información de traza y el acceso a otros datos relevantes de la ejecución como la pila o la memoria del procesador, se ha demostrado la capacidad de la información de traza para el diagnóstico de errores.

El IP se ha combinado con otras técnicas de tolerancia a fallos para construir técnicas híbridas de mitigación de errores:

- En primer lugar, se ha demostrado su uso en una implementación de ejecución simétrica en doble núcleo (*dual-core lockstep*) mediante la colaboración con la Universidad de Alicante, por la cual los dos núcleos de microprocesador ARM Cortex-A9 presentes en el dispositivo Zynq-7000 son configurados para ejecutar la misma aplicación y comparar sus resultados. El IP desarrollado en esta Tesis participa en esta técnica híbrida con la misión de comprobar que ambos procesadores ejecutaban el código en la región correcta de la memoria y con el comportamiento temporal adecuado. En el caso de que cualquiera de los dos núcleos muestre un comportamiento anómalo que ponga en riesgo la sincronización entre ambos, el IP lo indica mediante una señal de error de control de flujo.

- Por otra parte, el IP se ha utilizado para complementar una técnica de triplicación de datos mediante aceleración hardware utilizando la unidad NEON SIMD presente en el dispositivo Zynq-7000. El IP desarrollado en esta Tesis participa en esta técnica con la misión de comprobar que el microprocesador ejecuta el código en la región correcta de la memoria y con el comportamiento temporal adecuado. Además, también apoya al microprocesador realizando comprobaciones de consistencia sobre los datos triplicados para detectar errores.

Para validar el correcto funcionamiento del IP, se han seleccionado aplicaciones (*benchmarks*) representativas de distintas cargas de trabajo, como multiplicación de matrices, rutinas de ordenación o algoritmos de encriptación. El IP se ha validado mediante campañas de inyección de fallos mediante la técnica *Code Emulated Upset* (CEU) y también en radiación con protones de baja energía en el Centro Nacional de Aceleradores (CNA, Sevilla) y con neutrones en el Laboratorio Nacional de los Álamos (LANL, Estados Unidos). Adicionalmente, se han realizado campañas de inyección de errores mediante

láser para profundizar en la investigación en el diagnóstico de errores y complementar los resultados asociados en colaboración con la Universidad de Montpellier. Los resultados experimentales obtenidos en las distintas campañas de inyección y radiación demuestran su alta capacidad de detección y potencialidad en el diagnóstico de errores producidos por radiación.

Los resultados de esta Tesis han sido comunicados progresivamente a las conferencias más relevantes del sector: RADECS, NSREC y ESREF. Asimismo, se han logrado un total de siete publicaciones en revista, siendo cinco de ellas en la revista más prestigiosa del sector *IEEE Transactions on Nuclear Science* (TNS), JCR Q2; y dos más en la revista *Microelectronics Reliability* de Elsevier, JCR Q3.

Finalmente, el IP ha sido presentado en la conferencia *On-Board Data Processing* 2021 (OBDP 2021), un foro de orientación industrial dentro del ámbito de la industria espacial organizado por la Agencia Espacial Europea. En esta conferencia, el IP fue presentado como un producto disponible para ser introducido en el diseño de sistemas de procesamiento de aplicación espacial para incrementar su observabilidad y tolerancia a fallos.

El IP resultante de esta Tesis, desarrollada en el marco de un Doctorado Industrial entre la Universidad Carlos III de Madrid (UC3M) y la empresa Arquimea, se ha transferido satisfactoriamente al entorno empresarial en forma de un proyecto financiado por la Agencia Espacial Europea (ESA) para continuar su desarrollo y posterior explotación.

Palabras clave: Fiabilidad, Tolerancia a fallos, Fallos transitorios, Radiación, Detección de errores, Diagnóstico de errores, Mitigación de errores, SoC, COTS, ARM, Microprocesador, CoreSight, Interfaz de traza.

TECHNICAL SUMMARY

The miniaturization of the manufacturing technologies of integrated circuits and the associated increase in their performance have fostered in the last years an unprecedented development of electronic systems. Commercial electronic components, also known as Commercial-Off-The-Shelf (COTS) are present in a wide variety of devices commonly used in our daily life. Particularly, the use of microprocessors and highly integrated System-on-Chip (SoC) devices has fostered the advent of increasingly intelligent electronic devices which accelerate the progress of modern society. Microprocessors are present even in safety-critical systems, such as vehicles, planes, weapons, medical devices, implants, or power plants. In any of these cases, a fault could involve severe human or economic consequences, so the fault-tolerance becomes a requirement in their design and operation.

Fault-tolerance is a relevant subject in the space industry, as space vehicles such as satellites or spacecraft extensively integrate electronic systems and generally cannot be repaired once in-orbit. In space, radiation affect electronic components through cumulative effects such as Total Ionizing Dose (TID) or Displacement Damage Dose (DDD), which progressively deteriorate the functional parameters of integrated circuits until they eventually stop working properly. However, radiation also produces random effects known as Single-Event Effects (SEE), by that a single incident particle in the device can produce permanent faults (hard errors) or transient faults (soft errors). The effects produced by radiation in electronic devices have been investigated in detail since the space race and increasingly efficient hardening techniques have been developed to avoid, reduce, or mitigate their impact in the operation of systems in space.

During the second half of the 20th century, the space industry was mainly composed by publicly funded government agencies, such as NASA or the European Space Agency (ESA), who developed large space programs with the objective to avoid the risk at any expense, regardless of any cost or schedule restriction. This methodology intended to avoid the failure of the mission, which could compromise not only human lives, but also huge investments in the case of unmanned missions, and deteriorate the perception and the social support to space programs. Because of that, they only selected radiation-resistant components (known as rad-hard), specifically designed to be used in space through hardening techniques, capable to perform correctly under radiation. However, given their low market size and the associated extra development effort, rad-hard components have commonly a much higher associated cost and much lower performance that other equivalent COTS devices. In fact, COTS components have already been used successfully by NASA and ESA in missions that requested such high performance that could not be satisfied by any available rad-hard component.

In the recent years, a paradigm shift has been taking place in the space industry, fostered by the massive irruption of private companies that is facilitating the access to

space. Such companies generally seek objectives such as profitability and maximum benefit as providers of space-based services. In this context, the traditional methodology has been replaced by a new trend that pursues a trade-off between risk, cost, and schedule through risk management, in a paradigm known as "New Space". COTS components are more attractive for New Space companies, which are aimed to stand out in an increasingly competitive market, delivering the highest possible performance and reducing operational cost. Such objectives are generally not attainable through the solely use of rad-hard components, so there is an increasing interest in the use of COTS in space.

However, COTS components have not been specifically designed to be used in space and typically they include few specific techniques or resources to deal with the radiation effects in their operation. In addition, other factors related with the miniaturization of silicon technologies, such as the increase in integration density, the reduction of power supply voltages, and the shrinkage of noise margins, as well as the increment of the operating frequencies of electronic devices have led them to a higher vulnerability to radiation-induced soft errors. Such problem was in the past mainly associated to the space industry but has become a challenge for electronic systems performing safety-critical functions in avionics or even in terrestrial applications. Radiation-induced soft errors are considered to have the potential to become one of the major risks for the reliability of systems in space, and there is an increasing interest in their mitigation to extend the use of COTS components in future missions.

There is a wide range of existing hardening techniques to protect electronic devices against radiation or to mitigate its effects. Most of these techniques make use of redundancy schemes that consist in replicating the same structure or operation in multiple instances and comparing the results obtained by each replica. Rad-hard devices typically introduce hardware redundancy to mask or mitigate soft errors transparently to the user. However, it is not possible to apply this methodology in COTS components, as they are commercialized "as is", so they cannot be modified to improve their susceptibility to radiation effects. Moreover, the details about their internal architecture are usually unknown since the manufacturers do not reveal this information to protect their competitive advantage. Thus, there is a growing interest in the development of techniques allowing to understand and improve the behavior of COTS components under radiation without modifying their architecture and without interfering with their operation. Such techniques may facilitate the use of COTS components in space and maximize the performance of present and future space missions.

The development of fault-tolerance techniques for COTS microprocessors is a challenge. Historically, the memory buses have been used as an observation point, but this approach may not be feasible for hard-core microprocessors implemented in highly-integrated monolithic systems, as in the case of System-on-Chip (SoC) devices. This kind of devices generally present a limited number of available interfaces to observe the processor behavior. Observability is a key feature to understand the behavior of a device

in order to apply fault-tolerance techniques in an effective manner. The observability of a microprocessor is commonly reduced as its complexity and integration level increase.

In this Thesis, novel techniques have been developed to detect, diagnose, and mitigate radiation-induced errors in COTS microprocessors and SoCs with the objective of improving their fault-tolerance capabilities. As a main novelty, the developed techniques make use of the trace interface as an observation point, obtaining high compatibility with highly-integrated SoC devices. The trace interface is a resource commonly found in modern microprocessors, mainly intended to support software development and debugging activities during the design phase. Common tasks typically supported by the trace interface are code planning and code coverage checks. However, it is generally left unused during the operational phase of the system, so it can be reused with no cost. The trace interface constitutes a feasible connection point to observe microprocessor behavior in a non-intrusive manner and without disturbing processor operation.

As a result of this Thesis, an IP module has been developed capable to gather and decode the trace information of a modern, high-end, COTS microprocessor. The IP is highly configurable and customizable to support different applications and processor types. It has been designed in VHDL language and validated using the Xilinx Zynq-7000 device as a development platform, which is an interesting COTS device for the space industry. This device features a dual-core ARM Cortex-A9 processor, which is a good representative of modern, high-end, hard-core microprocessors. The resulting IP is compatible with the ARM CoreSight technology, which enables access to trace information in ARM microprocessors. The IP is able to decode the trace information produced by the Program Trace Macrocell (PTM) and the Instrumentation Trace Macrocell (ITM), both present in the Zynq-7000 device. Using the information obtained through the Trace Port Interface Unit (TPIU), the IP can detect errors in the execution flow of the microprocessor, including both cores present in the Zynq-7000 device, and in the application data, in real time and with very low latency. The IP includes every error detection and diagnosis techniques designed in this Thesis, which have been developed, tested, evaluated and integrated progressively:

- Control-flow error detection techniques:
  - Address range checking: The IP is capable of obtaining the successive program counter (PC) values from the trace information. The obtained PC values are continuously checked to assess whether the microprocessor is executing code inside a region of memory previously defined as valid. This region is configurable by the user through the configuration registers of the IP, used to delimit the allowed address ranges. In the case that the IP detects that the microprocessor is executing any instruction outside the allowed region, it is capable to detect it with very low latency and indicate it through an error signal. The IP can perform this check independently for each of the available microprocessor cores in the device.

- Address timing checking: Using the obtained program counter (PC) values, the IP can determine whether the processor is executing the application correctly in a timely manner. Using configuration registers, the user indicates a cyclic PC value in the program, which can be associated to its main loop. Whenever the IP detects than an excessive time, configurable by the user, has elapsed since the last complete execution of the main loop, it is indicated through an error signal. This technique is useful to detect with low latency any situation in which the processor may have lost control as a consequence of an error or an unexpected exception. The IP can perform this check independently for each of the available microprocessor cores in the device.

- Data error detection techniques:
  - Data range checking: The IP is capable of obtaining the value of the program data variables from the trace information. The obtained data values can be checked in any moment to evaluate whether they are valid inside the application according a predefined value range. The user can configure diverse value ranges as valid for certain variables using the configuration registers of the IP. Then, the IP checks automatically whether the value of the selected variables lies inside the allowed range and, if not, it indicates it through an error signal.
  - Data consistency checking: Using the obtained variable values of the program, the IP can check the consistency on duplicated or triplicated data. With this technique, the IP contributes to a performance improvement in the system as it offloads the microprocessor from performing this task. The IP can check automatically any comparison relation between groups of two and three data and indicates through an error signal the moment in which any of the expected relations is not met.

- Error diagnosis techniques:
  - The richness of the trace information goes beyond the sole purpose of error detection since, once the errors are detected, they can be also contextualized with the rest of the system operation. The contextualization consists in checking the observed error against the trace information which was generated immediately before it was detected. By analyzing this trace information and other relevant data about execution such as the processor stack or the memory contents, we have demonstrated the potential of the trace information for error diagnosis.

The IP has been combined with other fault-tolerance techniques to build hybrid error mitigation techniques:

- In one approach, we have demonstrated the use of the IP in a dual-core lockstep implementation, in collaboration with Alicante University. The two ARM Cortex-A9

microprocessor cores available in the Zynq-7000 device were configured to execute the same application and compare between their results. The IP developed in this Thesis was integrated in a hybrid technique to check the correct execution of both cores using address range checking and address timing checking. In the case that any of the cores showed a wrong behavior which could affect the synchronization between them, the IP indicated it with a control-flow error signal.

- In other approach, the IP was used to complement a data triplication technique which leveraged the NEON SIMD unit present in Zynq-7000 device to accelerate the replicated computations. The IP developed in this Thesis was integrated in a hybrid technique to check the correct execution of the microprocessor using address range checking and address timing checking. Additionally, the IP also supported the microprocessor by checking the consistency on the triplicated data to detect data errors.

To validate the correct operation of the IP, representative benchmarks of different workloads have been selected, such as matrix multiplication, sorting routines or encryption algorithms. The IP has been validated through fault injection campaigns, following the Code Emulated Upset (CEU) approach, and also under proton and neutron irradiation campaigns, performed in *Centro Nacional de Aceleradores* (CNA, Seville, Spain) and Los Alamos National Laboratory (LANL, EEUU). Additional laser fault injection campaigns have been performed in collaboration with the University of Montpellier to deepen in the error diagnosis research and complement the associated results. The experimental results obtained in the different fault injection and irradiation campaigns demonstrate the high detection capabilities of the developed IP and its potential in the diagnosis of radiation-induced errors.

The outcomes of this Thesis have been communicated progressively to the most relevant conferences in the field, namely RADECS, NSREC and ESREF. In addition, a total of seven journal publications have been attained, being five of them in the most renowned journal in the field, namely IEEE Transactions on Nuclear Science (TNS), JCR Q2; and two in the Microelectronics Reliability journal, by Elsevier, JCR Q3.

Finally, the IP has been presented in the On-Board Data Processing 2021 (OBDP 2021) conference, which is an industry-oriented forum in the space sector, organized by the European Space Agency. In this conference, the IP was presented as a product available to be introduced in the design of space-oriented processing systems, to increment their observability and fault-tolerance capabilities.

The resulting IP of this Thesis, developed in the framework of an Industrial Ph.D. between the University Carlos III of Madrid (UC3M) and the company Arquimea, has been successfully transferred to the company business as a project sponsored by European Space Agency to continue its development and subsequent commercialization.

Keywords: Reliability, Fault-tolerance, Soft errors, Radiation, Error detection, Error diagnosis, Error mitigation, SoC, COTS, ARM, Microprocessor, CoreSight, Trace interface.

LIST OF ABBREVIATIONS

AES Advanced Encryption Standards
AI Artificial Intelligence
ALU Arithmtic Logic Unit
AP All Programmable
API Application Programming Interface
ARM Advanced Risc Machines
ASIL Automotive Safety Integrity Level
AVF Architectural Vulnerability Factor
AXI Advanced eXtensible Interface

BB Basic Block
BER Backward Error Recovery
BFI Branch-Free Interval
BIST Built-In Self Test
BOX Buried OXide
BSP Board Support Package

CCA Control flow Checking approach using Assertions
CEDA Control-flow Error Detection through Assertions
CEU Code Emulated Upset
CFC Control-Flow Checking
CFCSS Control Flow Checking by Software Signatures
CFE Control-Flow Error
CFG Control-Flow Graph
CMOS Complementary Metal-Oxide-Semiconductor
CMP Chip Level Multiprocessor
CNA Centro Nacional de Aceleradores
COTS Commercial Off-The-Shelf
CPU Central Processing Unit
CUT Code Under Test

DA Data Abort
DCLS Dual-Core LockStep
DDD Displacement Damage Dose
DDR Double Data Rate
DE Data Error
DfT Design for Test

DICE	Dual Interlocked storage CEll
DM	Data Memory
DMA	Diseño Microelectrónico y Aplicaciones
DMR	Dual Modular Redundancy
DMT	Duplex Multiplexed in Time
DRAM	Dynamic Random Access Memory
DT2	Dual Duplex Tolerant to Transients
DUE	Detected Unrecoverable Error
DUT	Device Under Test
ECCA	Enhanced High level Control flow checking approach using Assertions
EDAC	Error Detection And Correction
EDDI	Error Detection by Duplicated Instructions
EE	Exception Error
ELDRS	Enhanced Low Dose Rate Sensitivity
EMIO	Extended Multiplexed Input-Output
ESA	European Space Agency
ESREF	European Symposium on Reliability of Electron Devices, Failure Physics and Analysis
ESTEC	European Space Research and Technology Centre
ETB	Embedded Trace Buffer
ETM	Embedded Trace Macrocell
FER	Forward Error Recovery
FF	Flip Flop
FIFO	First-In, First-Out
FMC	FPGA Mezzanine Card
FP	Frame Pointer
FPGA	Field Programmable Gate Array
FTM	Fabric Trace Macrocell
GCR	Galactic Cosmic Rays
GPIO	General Purpose Input Output
HDL	Hardware Description Language
HETA	Hybrid Error-Detection Technique Using Assertions
HPSC	High-Performance Spaceflight Computing
I/O	Input / Output
IEEE	Institute of Electrical and Electronics Engineers

IES	Institut d'Electronique et des Systèmes
IP	Intellectual Property
ITM	Instrumetntation Trace Macrocell
JCR	Journal Citation Report
JPL	Jet Propulsion Laboratory
JTAG	Joint Test Action Group
LANL	Los Alamos National Laboratory
LANSCE	Los Alamos Neutron Science Center
LED	Light Emitting Diode
LEO	Low Earth Orbit
LET	Linear Energy Transfer
LR	Link Register
LUT	Look-Up Table
MBU	Multiple Bit Upset
MCA	Machine Check Architecture
MCU	Multiple Cell Upset
MCU	MicroController Unit
MIO	Multiplexed Input-Output
MMU	Memory Management Unit
MOS	Metal-Oxide-Semiconductor
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
MP	Multiprocessor
Mth	Master thread
NASA	National Aeronautics and Space Administration
NMOS	Negative-channel Metal-Oxide-Semiconductor
NMR	N-Modular Redundancy
NSREC	Nuclear and Space Radiation Effects Conference
OBDP	On-Board Data Processing
OCD	On-Chip Debugging
OCM	On Chip Memory
OS	Operating System
OSIP	Open Space Innovation Platform
PA	Prefetch Abort
PC	Program Counter

PDTC Program and Data Trace Checker
PFT Program Flow Trace
PL Programmable Logic
PM Program Memory
PMOD Peripheral MODules
PMOS Positive-channel Metal-Oxide-Semiconductor
POSIX Portable Operating System Interface
PS Processing System
PTM Program Trace Macrocell
PUM Processor Under Monitoring

rad radiation absorbed dose
RADECS Radiation and its Effects on Components and Systems
RHA Radiation Hardness Assurance
RHBD Radiation-Hardening by Design
RHBP Radiation-Hardening by Process
RMT Redundant MultiThreading

SAA South Atlantic Anomaly
SB Static Block
SBC Single-Board Computer
SD Secure Digital
SDC Silent Data Corruption
SDK Software Development Kit
SEB Single-Event Burnout
SECDED Single Error Correction, Double Error Detection
SEE Single-Event Effect
SEFI Single-Event Functional Interrupt
SEGR Single-Event Gate Rupture
SEL Single-Event Latchup
SEM Soft Error Mitigation
SER Soft Error Rate
SERESSA School on the Effects of Radiation on Embedded Systems for Space Applications
SES Single-Event Snapback
SET Single-Event Transient
SEU Single-Event Upset
SFI Statistical Fault Injection
SIHFT Software Implemented Hardware Fault Tolerance
SIMD Single-Instruction Multiple Data

SMT Simultaneous Multithread
SoC System-on-Chip
SoI Silicon on Insulator
SoR Sphere of Replication
SPCD Selective Procedure Call Duplication
SPENVIS SPace ENVironment Information System
SPI Serial Peripheral Interface
SPMD Single Program Multiple Data
SRAM Static Random Access Memory
Sth Shadow thread
STI Shallow Trench Isolation
STM System Trace Macrocell
SW SoftWare
SWaP Size, Weight and Power
SWIFT SoftWare-Implemented Fault-Tolerance
SWIT SoftWare Instrumentation Trace

TCLS Triple-Core LockStep
TID Total Ionizing Dose
TMR Triple Modular Redundancy
TNS Transactions on Nuclear Science
TPIU Trace Port Interface Unit
TRL Technology Readiness Level

UC3M University Carlos III of Madrid
UI Undefined Instruction
USB Universal Serial Bus

VHDL VHSIC Hardware Description Language

YACCA Yet Another Control Flow Checking Approach

ZYBO ZYnq BOard

CONTENTS

AGRADECIMIENTOS iii
PUBLISHED AND SUBMITTED CONTENT iv
OTHER RESEARCH MERITS vi
RESUMEN NO TÉCNICO vii
NON-TECHNICAL SUMMARY x
RESUMEN TÉCNICO xii
TECHNICAL SUMMARY xviii
ABBREVIATIONS xxiv
CONTENTS xxix
LIST OF FIGURES xxxv
LIST OF TABLES xxxvii
1. INTRODUCTION 1
1.1. Motivation 1
1.2. Industrial Ph.D. supported by the Community of Madrid 3
1.3. Objectives 3
1.4. Document structure 5
2. DEPENDABILITY UNDER RADIATION IN COTS MICROPROCESSORS . . 7
2.1. Dependability in space applications 7
2.1.1. Dependability attributes 7
2.1.2. Faults, errors, and failures 8
2.1.3. Dealing with faults 9
2.1.4. Dependability threats in spacecraft 11
2.2. Radiation effects on electronics 12
2.2.1. The space environment 12
2.2.2. Cumulative effects 14
2.2.3. Single-event effects 16
2.3. Addressing radiation effects 20
2.3.1. Radiation hardening 20

2.3.2. Component testing 26
2.3.3. Component qualification 34
2.4. Trends in space industry 35
2.4.1. Background: the so-called "traditional space" 35
2.4.2. New Space trends 36
2.4.3. Use of COTS in space applications 37
2.4.4. COTS testing and qualification 38
2.5. Microprocessors under radiation 41
2.5.1. Microprocessor errors 41
2.5.2. Microprocessor testing. 42
2.6. Microprocessor fault-tolerance 46
2.6.1. Fault-tolerance techniques for microprocessors 46
2.6.2. Fault-tolerance techniques for COTS microprocessors 56
3. MATERIALS AND METHODS 70
3.1. Resources 70
3.1.1. Vehicle of study 70
3.1.2. Facilities 72
3.1.3. Fault injection tool 75
3.1.4. Other equipment 75
3.2. Methodology 76
3.2.1. Chronology 78
3.2.2. Development 78
3.2.3. Validation 81
3.2.4. Industrialization 83
3.2.5. Dissemination 84
4. PTM-BASED HYBRID ERROR-DETECTION ARCHITECTURE FOR ARM MICROPROCESSORS 86
Abstract 86
4.1. Introduction 86
4.2. Related work 88

4.3. Hybrid architecture. 90
4.3.1. Hardware monitor 90
4.3.2. Data error detection 92
4.4. Experimental results 92
4.4.1. Experimental setup. 92
4.4.2. Proton irradiation. 93
4.4.3. Fault injection 94
4.5. Conclusions and future work 95
Acknowledgements 96
References 96
5. ONLINE ERROR DETECTION THROUGH TRACE INFRASTRUCTURE IN ARM MICROPROCESSORS 99
Abstract. 99
5.I. Introduction 99
5.II. Related work 101
5.III. Trace-Based Error Detection Approach. 103
5.III-A. CoreSight Subsystem 104
5.III-B. Program and Data Trace Checker. 105
5.IV. Experimental results 107
5.IV-A. Experimental setup. 107
5.IV-B. Radiation results 108
5.IV-C. Trace information analysis 111
5.V. Conclusion 113
Acknowledgement 113
References 114
6. DUAL-CORE LOCKSTEP ENHANCED WITH REDUNDANT MULTITHREAD SUPPORT AND CONTROL-FLOW ERROR DETECTION . . . 117
Abstract. 117
6.1. Introduction. 117
6.2. Proposed lockstep approach. 119
6.2.1. Architecture. 119

6.2.2. Data error detection 120
6.2.3. Control-flow error detection 122
6.3. Experimental results 123
6.4. Conclusions 126
Acknowledgements 126
References 126
7. THE USE OF MICROPROCESSOR TRACE INFRASTRUCTURES FOR RADIATION-INDUCED FAULT DIAGNOSIS 129
Abstract 129
7.I. Introduction 129
7.II. Microprocessor Fault Diagnosis 131
7.III. Fault Diagnosis Approach 132
7.III-A. Trace Analysis 135
7.III-B. Memory and Stack Analysis 136
7.IV. Experimental Results 136
7.IV-A. Common Experimental Setup 136
7.IV-B. Neutron Irradiation Results 137
7.IV-C. Proton Irradiation Results 140
7.V. Conclusion 145
Acknowledgement 146
References 146
8. ERROR DETECTION AND MITIGATION OF DATA-INTENSIVE MICROPROCESSOR APPLICATIONS USING SIMD AND TRACE MONITORING 149
Abstract 149
8.I. Introduction 149
8.II. Background and Related Work 151
8.III. Proposed Approach 153
8.III-A. Data Hardening With NEON SIMD Coprocessors 153
8.III-B. Trace Monitoring 157
8.IV. Experimental Results 159
8.IV-A. Performance Comparison 161

8.IV-B. Neutron Radiation Results. 162
8.V. Conclusion 164
Acknowledgement 165
References 165
9. MICROPROCESSOR ERROR DIAGNOSIS BY TRACE MONITORING UNDER LASER TESTING 168
Abstract 168
9.I. Introduction 168
9.II. Related Work 169
9.III. Evaluation Methodology 171
9.III-A. Device under test 171
9.III-B. Software case study 172
9.III-C. Trace monitoring approach 172
9.III-D. Laser fault injection 173
9.III-E. Relation to previous work 176
9.IV. Experimental Results and Discussion 176
9.V. Conclusion 184
Acknowledgement 184
References 184
10. IP TO DETECT AND DIAGNOSE ERRORS IN COTS MICROPROCESSORS THROUGH THE TRACE INTERFACE 187
Abstract 187
10.1. Introduction 187
10.2. Microprocessor Error Detection and Diagnosis 188
10.3. Error Detection and Diagnosis IP 191
10.3.1. Interface description 192
10.3.2. Functional description 193
10.3.3. Checking resources 194
10.3.4. Error diagnosis 196
10.4. Applications 196
10.5. Conclusions 198

Acknowledgements . . . 199
References . . . 199
11. CONCLUSIONS AND FUTURE WORK . . . 202
11.1. Conclusions . . . 202
11.2. Future work . . . 207
BIBLIOGRAPHY . . . 209

LIST OF FIGURES

3.1 Experimental setup at CNA 73
3.2 Experimental setup at LANSCE 74
3.3 Experimental setup at Montpellier University 75
3.4 RaspberryPi connected to ZYBO board for test campaigns 82

4.1 Proposed system overview 90
4.2 Experimental setup overview 92
4.3 Errors by register 96

5.1 General trace-based error detection architecture 103
5.2 Code instrumentation examples and data checker operation 106

6.1 Dual-Core LockStep architecture 120
6.2 Redundant threaded matrix multiplication 122
6.3 Register sensitivity to errors 125

7.1 Trace generation and processing architecture 134
7.2 Trace simulation test bench 135
7.3 Bar graph of diagnosed addresses for 128×128 matrix size under neutron irradiation 138
7.4 Bar graph of diagnosed addresses for 32×32 matrix size under neutron irradiation 139
7.5 Bar graph of diagnosed addresses for 32×32 matrix size under proton irradiation 141
7.6 Exception Errors versus Data Errors 142
7.7 Memory errors causing Data Errors 143
7.8 Memory errors captured by the trace 144
7.9 Computation errors captured by the trace 145

8.1 NEON register file and instruction example 154
8.2 SIMD-based hardened data 154

8.3 Code example of data hardening using NEON data types 156
8.4 PDTC architecture 159
8.5 Experimental setup 160
8.6 Cross section bar graph 164

9.1 Experimental setup detail 174
9.2 Infrared microphotograph of the processing system of the DUT showing the fault injection zones 174
9.3 Block diagram of the experiment 175
9.4 Error histogram for Z1 (CUT1) - L1 Data Cache 177
9.5 Error histogram for Z2 (CUT1) - L1 Instruction Cache 178
9.6 Error histogram for Z4 (CUT1) - L2 Cache 178
9.7 Error histogram for Z5 (CUT3) - OCM 179
9.8 Error histogram for Z3 - unidentified registers region 180
9.9 Last instructions before Data Aborts related to L2 Cache (CUT1) 182
9.10 Error geometrical distribution (CUT1) 183
9.11 Error geometrical distribution by category (CUT1) 183

10.1 Top level view of the IP and interfaces 192
10.2 Internal architecture view of the IP 194
10.3 IP integrated in a) binary and b) ternary architecture configuration 198

LIST OF TABLES

4.1 Experimental results of proton irradiation 94
4.2 Experimental results of fault injection 95

5.I Synthesis Results 106
5.II Radiation Results: Observed Errors 109
5.III Radiation Results: Cross Section 111

6.1 Injection campaign results 124

7.I Detailed Address Contents 138
7.II Data Error Diagnosis 139

8.I Performance Overhead 161
8.II Neutron Radiation Results 163

9.I Zones used for laser fault injection 175
9.II PC Progression to Invalid Value 180
9.III Detailed Address Contents 181

10.1 IP specifications 196

1. INTRODUCTION

Improvements in silicon-based technologies have fostered a great development in electronics industry. Throughout the last years, electronic devices and, particularly, microprocessors, have become smaller, faster, more powerful and more efficient, resulting attractive for an increasing number of applications. Integration density has also reached unprecedented levels, allowing to implement heterogeneous systems including multiple processor cores, mixed-signal peripherals and programmable logic in a single chip, conforming which is known as System-on-Chip (SoC) devices.

Microprocessors are widely used in everyday commercial appliances, as in smartphones, tablets or personal computers. However, they are also increasingly common in systems performing critical tasks, such as automated fabrication, personal data management, financial operations, or even safety-critical applications like emergency systems, avionics, or autonomous driving. Electronics are extensively used in space-oriented systems, and microprocessor-based applications are commonly introduced in spacecraft and payload control and management.

Users may expect that systems continuously operate correctly. However, any system is subject to have faults. Faults in space systems must be minimized as spacecraft are mostly unfeasible to repair. Once a fault occurs, detection and diagnosis are desirable capabilities to minimize consequences and perform efficient corrective actions. Dealing with faults is important to maximize the lifespan of a system, however, doing it at a minimal cost is a challenge. Methodologies are constantly evolving to efficiently reduce the impact of faults in systems in a cost-effective manner. This is particularly relevant to maximize benefit in an increasingly competitive space industry.

1.1. Motivation

Faults may affect the operation of any system, undesirably compromising the quality of the delivered service. When affecting microprocessor operation, faults may result in a degraded user experience in commercial electronic appliances. However, when affecting safety-critical systems, the presence of faults can produce severe consequences that ultimately may produce damages, involve economic losses or endanger human lives. In space, radiation is a major source of faults in electronic integrated circuits and, if not properly handled, may derive in partial or complete mission loss.

Systems performing critical applications must typically satisfy high reliability and availability requirements, even when operating in harsh environments. The quality of a system to perform correctly in the presence of faults is called fault tolerance, and it is a desirable feature of critical systems. In the case of space-oriented systems, maintenance

after deployment is commonly unfeasible or completely unavailable. Moreover, the space environment is challenging for electronics to operate in due to radiation exposure and its various undesirable effects on integrated circuits. For that reason, fault tolerance becomes a crucial issue to be addressed when developing successful space missions.

In the last decades, the space industry has developed technologies and techniques to deal with radiation effects in electronics and to test their effectiveness. Integrated circuits capable of operating correctly in the presence of radiation are commonly known as radiation hardened or rad-hard and can be qualified for their use in space through radiation testing. However, radiation hardened integrated circuits typically present disadvantages such as higher power consumption, higher cost and lower performance than non-hardened counterparts. Regarding rad-hard microprocessors, they commonly lag two or three generations behind commercially available ones, known as Commercial-Off-The-Shelf, (COTS). In fact, available radiation hardened solutions for space applications present excessive cost and lack of performance to enable the next generation of space missions.

Size, weight and power are key design drivers in space missions, as they are restricted throughout the mission, and electronic components are selected accordingly. Additional considerations such as radiation hardness and performance are also taken into account to fulfill mission requirements and achieve mission objectives. In the case that no available rad-hard device would meet the requirements of a space mission, the entire mission could be affected, introducing planning delays and other deviations. This is not uncommon, as space mission designers tend to seek for high computational on-board capabilities in spacecraft that may enable ambitious scientific, exploration or commercial purposes. In the last years, new actors such as research entities and private companies have irrupted in the space industry, which was traditionally composed by government agencies, configuring a new tendency which is known as New Space. New Space introduces complementary requirements on traditional spacecraft design such as limited budgets and constrained schedules, typically combined with low power, reduced weight and high performance features. In the case that available rad-hard devices do not provide the expected performance or are not affordable for a given mission, the use of COTS may be considered, given that protection against radiation effects can be provided.

Commercial-Off-The-Shelf devices are those conceived to be used in ground-based applications, such as domestic devices. They sometimes present a reduced set of built-in fault-tolerance mechanisms, mainly in the case of devices designed for industrial and autonomous driving applications. However, COTS devices are not designed for its use in space, so they have not deliberately been radiation-hardened and, commonly, present an unknown susceptibility to radiation effects. Nevertheless, the suitability of a COTS device for a space mission can be evaluated through radiation testing and, if it is found to fulfill mission requirements, it would be included in the design. If necessary, a COTS device could be hardened by introducing additional features at device, circuit or system level. For microprocessors, techniques can be applied both on the software and the hardware. However, hardware techniques present limitations as hardware typically cannot be modified

since the architecture of COTS devices is commonly unknown. This process is usually only performed when the required performance is unattainable by any available rad-hard component as the COTS qualification process is usually long, complex and expensive. For this reason, new techniques to harden COTS under radiation are continuously under research to deal with COTS observability, testability and hardening challenges.

This work explores new ways to facilitate the use of SoCs based on COTS microprocessors in space, by providing radiation-induced error detection and diagnosis capabilities in a non-intrusive way using externally available resources and interfaces. A high-performance, multi-core commercial SoC based on ARM processor architecture was used as a vehicle of study of the proposed techniques. ARM devices present deep penetration in commercial electronics industry and are also of interest in space industry. Obtained results would be applicable in almost any high-reliability area and, with special relevance, in space applications.

1.2. Industrial Ph.D. supported by the Community of Madrid

This Thesis was developed in the framework of an industry-academia collaboration between the aerospace company Arquimea and University Carlos III of Madrid (UC3M).

The designation of Industrial Ph.D. applies to programs whose candidates participate in industrial experimental development or research work projects connected to their theses. Industrial Ph.D. programs involve effective collaboration between a company and a research entity to promote technology transfer and industrial applications of research results.

This work has been supported in part by the Community of Madrid under Grant IND2017/TIC-7776 awarded in 2017 within a call for supporting the realisation of Industrial Ph.D. programs.

1.3. Objectives

The fundamental objective of this work is to explore new error mitigation solutions for Sytems-on-Chip (SoCs) based on Commercial-Of-The-Shelf (COTS) microprocessors to enable the use of higher-performance and lower-cost processing systems in space applications. In this kind of devices it is usually not feasible to modify the hardware to implement fault tolerance techniques, so they must be achieved by other means. For this purpose, it is considered essential in this work to leverage existing infrastructures and interfaces already available in microprocessors. Such infrastructures and interfaces may be used to externally check correct circuit operation and harden it in a non-intrusive manner. This work focuses in non-intrusiveness and simplicity to minimize negative impacts of error mitigation in area and performance, to obtain a low-footprint solution to harden the processor without disturbing it.

It is remarkable the case of trace and debug infrastructures among others to be used for error mitigation purposes and, particularly, the trace interface. Trace and debug infrastructures are designed to support application debug during software development phase, and are provided by most modern processors for this purpose. Once the application has been deployed, such infrastructures are no longer needed and are left unused, so it is possible to reuse them for a different purpose. In particular, the trace interface provides traceability of the executed code and associated program data, observing processor behavior without interfering in execution. Information provided by the trace interface can be used to detect errors in processors and has already been used for that purpose on soft-core processors by the advisors of this Thesis at the *Diseño Microelectrónico y Aplicaciones* (Microelectronic Design and Applications, DMA) research group in UC3M [1]–[8]. However, such approach has not been yet attempted in more complex, high performance, hard-core processors.

The ARM technology used as a vehicle of study presents relevant elements in its architecture commonly found in high-end processing systems that are attractive for this work. The Zynq-7000 device family provided by Xilinx has been selected as development platform as it includes multiple hard-core high-performance ARM processors as well as programmable logic in a single chip. This SoC configuration provides flexibility and conveniently supports the development and testing of both the processors and the checking techniques.

More specifically, the objectives of this work are the following:

I Development of non-intrusive error mitigation techniques for COTS processors based on the trace interface, focusing in both control-flow errors and data errors.

II Development of diagnosis techniques for processor errors based on the information provided by the trace infrastructures.

III Validation of the designed techniques by fault injection and irradiation campaigns. Fault injection is considered the most suitable approach for reliability evaluation during the preliminary steps. Nevertheless, radiation experimental results are essential for the acceptance of results by the scientific and technical community, as they can accurately reproduce radiation effects. This objective is devoted to demonstrate the suitability of the proposed techniques in a representative environment.

IV Implementation of the developed techniques for space application circuits. As an Industrial Ph.D., the ultimate objective of this work is to obtain a technology susceptible to commercialization and identify how to industrialize it.

This work is ultimately oriented to increase the market penetration of Arquimea company. Arquimea is currently a European reference in the design, manufacturing and testing of space-oriented integrated circuits. Through this work, the company will make

available a proprietary technology for robust SoC design that provides a competitive advantage in the sector.

1.4. Document structure

This Thesis document has been elaborated by compendium of publications modality. The structure of the document is as follows:

- Chapter 2 introduces the relevant concepts within this document and summarizes the state of art for this work.
- Chapter 3 reviews the materials and methods used in this work.
- Chapter 4 contains the publication [J1]. In this publication a hardware module is proposed as a preliminary proof of concept for control-flow error detection through the trace interface, and is validated by fault injection and irradiation campaigns, initiating the accomplishment of objectives I and III.
- Chapter 5 contains the publication [J2]. In this publication, the proof of concept module is extended for both control-flow and data error detection and is validated by irradiation campaigns, going deeper in objectives I and III. Error diagnosis is firstly proposed in this publication, settling the basis for attaining objective II.
- Chapter 6 contains the publication [J3]. In this publication, the developed fault detection techniques are applied to a more complex fault-tolerant system implementing hardware redundancy by using both available cores in the Zynq-7000 platform in lockstep configuration. By the combination of a set of software and hardware techniques in collaboration with Alicante University, including the developed hardware module, a hybrid fault-tolerant system is proposed and tested under fault injection, accomplishing objectives I and III.
- Chapter 7 contains the publication [J4]. In this publication, the error detection capabilities of the hardware module are explored for the purpose of error diagnosis. The proposed system is evaluated under fault injection and irradiation campaigns, demonstrating fine granularity at error classification and richness of information to perform error diagnosis, accomplishing objective II.
- Chapter 8 contains the publication [J5]. In this publication, the hardware module is used to detect control-flow errors combined with a data error mitigation technique using existing SIMD hardware acceleration resources in the processor architecture. The resulting hybrid approach is validated under irradiation campaigns, showing up to 99.9% error coverage. With the work presented in chapters 6 and 8, the objective I is considered fully accomplished.

- Chapter 9 contains the publication [J6]. In this publication, the error diagnosis approach is tested under laser fault injection in collaboration with Montpellier University, providing additional information about fault location and timing. Results demonstrate high potential in diagnosis capabilities through the trace information using the proposed hardware module. With the work presented in chapters 7 and 9, the objective II is considered fully accomplished.

- Chapter 10 contains the publication [C1]. In this publication, the developed hardware module is presented as a commercially available IP product at an industrial forum in the European space sector. This work demonstrates the efforts performed by Arquimea for the effective transference and commercialization of the developed technology, accomplishing objective IV.

- Chapter 11 outlines the conclusions of this Thesis and proposes future developments based on this work. Not only the main achievements in this work are remarked, but also a new project with European Space Agency (ESA) based on this Thesis to further develop and test the IP is announced in this chapter. With the achievement of this contract, the objective IV is considered fully accomplished.

2. DEPENDABILITY UNDER RADIATION IN COTS MICROPROCESSORS

In this chapter, an overview of the basic concepts addressed in this Thesis is included. Despite chapters 4 to 10 have each one their own dedicated sections for state of the art and related work, the present chapter is willing to unify them in a single entity and to provide a complete landscape of the state of the art related to the whole work in a comprehensive manner. This chapter also presents general concepts that are not described in further chapters to ease understanding of this work by readers that are not familiar with either of the topics, and to better contextualize this work in the current research trends.

2.1. Dependability in space applications

Critical applications are built around systems that must not fail, which is the case of space applications. In the case of unmanned space missions, most of them cannot be repaired once in-orbit, so an unexpected failure would lead to mission loss. Dependability is a key attribute in critical systems design, and is associated with the quality of service that a particular system provides [9]–[11].

Dependability is defined as "the ability of a system to deliver service that can justifiably be trusted" [12]. As it is defined in broad terms, there is not a unique approach to attain dependability, but different approaches are valid depending on the circumstances. Finding the proper implementation of this concept is key to support advanced society in many fields. In particular, space applications rely on dependability concepts to survive and protect valuable information from the harsh conditions of outer space.

This section will introduce a basic overview of dependability concepts and illustrate their application to the space challenges.

2.1.1. Dependability attributes

The justification of trust is crucial to effectively attain the dependability requirements of a system. Trust can be defined as accepted dependence from the point of view of a system in a context in which the dependability of other system can affect the dependability of the first one. Consequently, dependability can also be defined as the ability of a system "to avoid service failures that are more frequent and more severe than is acceptable" by other systems [10].

Dependability is usually described as a combination of attributes, including the following [9]:

- Reliability is considered the "continuity of correct service" [10]. It is defined as the probability that a system delivers the correct service throughout a complete interval of time [11].
- Availability is the "readiness for correct service" [10]. It is defined as a statistical parameter as the probability that a system is capable to correctly provide service at the moment it is requested [11].
- Maintainability is the "ability to undergo modifications and repairs" [10]. It is defined as a statistical parameter as the probability that a system can be restored to provide correct service once it has failed [11].
- Testability is "the ability to test for certain attributes within a system" [9]. A system with high testability will provide easy means to identify the causes of failures, as for example automated testing.
- Performability can be considered as "performance-related measures of dependability" [12]. It is defined as a statistical parameter as the probability that a system will perform at some certain level (or above) at a given instant of time. Performability differs from availability and reliability in that those are measures for all system functions to be performed correctly, while performability only considers a certain subset of functions to be performed correctly. Performability is linked to the concept graceful degradation, which "is the ability of a system to automatically decrease its level of performance to compensate for hardware and software faults" [9]. In most applications, and especially in space missions, the option to lose some non-critical features and maintain the important ones available, as communication subsystems, is more attractive than the option to get the whole system unavailable, that would lead to mission loss.
- Safety is the "ability of a system to show a safe behavior in the presence of a fault that would lead to an unacceptable failure" [11]. It is defined as a statistical parameter as the probability that a system will either perform its functions correctly or, alternatively, will discontinue its functions in such a way that does not disrupt the operation of other systems or compromise the health of any people associated with the system [9]. A system is safe as long as it guarantees that its operation will not present catastrophic consequences to the user and the environment [10].
- Finally, survivability is the "capability of a system to fulfill its mission in a timely manner" [12]. In other words, is the capability of a system to remain operational and perform its function during the required amount of time.

2.1.2. Faults, errors, and failures

The words fault, error and failure have similar meanings. However, these terms must be very well established when concerning dependability as they represent different concepts.

A fault is "a physical defect, weakness, imperfection, or flaw that occurs within some hardware or software component" [9] that may be the "adjudged or hypothesized cause of an error" [10]. Faults can be categorized according to different criteria [10], [12], to identify their origin (internal or external to the system), their domain (affect the software or the hardware of the system), or their persistence (as can be permanent, intermittent or transient).

"An error is the manifestation of a fault" [9]. Specifically, an error is the "deviation of the internal state of a system from the expected" [11] "that may lead to its subsequent service failure" [10]. When the presence of an error is indicated by a signal or a message, it is said that it is detected. Errors that are present but undetected are called latent errors [10]. In the dependability analysis, an error is considered an intermediate, but necessary, step between a fault and a failure.

"A failure occurs when an error is propagated to the service interface and unacceptably alters the service delivered by the system" [12]. Failures can be categorized according to different criteria [10], [12] to determine their extent (a subset or all provided services are affected), their domain (the service is affected in the delivered content and/or the delivered timing that is provided), their controlability (the behavior of the failed system is according to the specification or is unexpected) or their consequences (that may be minor, when their cost is comparable to the benefit of correct service or catastrophic, when their cost is much higher than the benefit).

Consequences of failures drive the development of risk analysys to determine the hazards and consequent accidents that can occur during system operation. "Risk is a measure of the dangerousness of an accident. It is given by the product of the frequency (or probability of occurrence) and the severity of an accident, commonly termed risk probability and risk severity" [11].

A system can fail in different ways, which are called failure modes. The consequences and risks associated to each failure mode are different and can be classified according to failure severities. The severity of a failure is quantified by evaluating its impact on dependability attributes, such as the failure duration or the threats to human lives [10].

It is remarkable that the cause-effect relation between faults, errors and failures is not finished when a failure occurs. Not only faults provoke errors and errors provoke failures, but also failures can trigger new faults on other systems by interaction or interference processes [10].

2.1.3. Dealing with faults

Faults are intrinsic in any system development and operation. The contribution of faults to the behavior of a system is a risk that must be accepted and tackled at the beginning of the design process. A widely accepted view about the influence of faults in systems is the so-called bathtub curve [13], [14] . The curve presents steep sides and a flat bottom,

forming the shape a bathtub. The bathtub curve describes the failure probability of a system during its lifetime by the sum of contributions of three failure functions, producing three different phases. The first part of the graph comprises the failures which appear early in the system lifetime, known as early failures, produced mostly by faults that were not detected by quality tests. As the units presenting early failures can be identified and discarded, the amount of early failures decreases with the time, this situation is also referred to as infant mortality. The second part of the curve is a flat probability of failures over time representing the contribution of random, difficult to predict effects, which are known as random failures. The duration of the second phase is related to the expected lifetime of the system. Finally, the third part of the graph introduces the effects of aging and wear-out in the materials or other components of the system that produce an increment of the failure rate with the time, known as wear-out failures.

By providing means to identify and analyze the risks associated to a system, approaches can be implemented to mitigate them and, eventually, attain dependability requirements with high trust [9]. Mainly, four mitigation approaches can be considered: fault forecasting, fault prevention, fault removal and fault tolerance:

- "Fault forecasting means to estimate the present number, the future incidence, and the likely consequences of faults" [10]. The result is "qualitative evaluation to identify, classify and rank failure modes and calculate the probability to satisfy dependability attributes" [12].
- "Fault prevention, also called fault avoidance, means to prevent the occurrence or introduction of faults" [10] in the system. "Fault prevention is typically attained by quality control techniques" [12] that are applied across the design process. Such techniques include "design reviews, component screening, testing, and other quality control methods"[9]
- "Fault removal means to reduce the number and severity of faults" [10]. It "can be applied during development in three steps: verification, diagnosis, correction; or during operation with corrective or preventive maintenance" [12].
- "Fault tolerance means to avoid service failures in the presence of faults" [10]. "Fault tolerance is an attribute that is designed into a system to achieve some design goal, just as a design must meet many functional and performance goals" [9]. "The factor that distinguishes fault tolerance from maintenance is that maintenance requires the participation of an external agent" [10] while fault tolerance is a built-in capability. Two strategies [10] are commonly followed to attain fault-tolerance exploiting the introduction of redundancy in the system that can be active and passive [11]:
  - Error detection and recovery. Once a fault, or an error, has been identified in the system, it is located and an action is performed to either remove the fault to continue normal operation, or to contain its consequences, usually by reducing the system performance [11].

- Fault masking: the redundancy previously introduced in the system prevents the fault to produce errors. In other words, the fault is masked and the recovery step is not needed [9], [11].

2.1.4. Dependability threats in spacecraft

Spacecraft are complex systems composed of different subsystems tightly coupled together to fulfill a mission. Not only electronic devices are included in spacecraft, but also mechanical, electric or fluid-based subsystems are involved. Examples of those subsystems are communications, power management, thermal management, orientation management, propulsion, instrumentation and payloads.

Spacecraft systems "use microelectronics for command and control functions, for signal acquisition and processing functions, and for data storage" [15]. Microelectronics aboard spacecraft experiences and adverse and extreme environment, which becomes a threat due to different factors [16]:

- Mechanical stress, including extremely intense vibration and high acceleration during launch and, eventually, landing phases. The different gravity conditions in other planets or celestial bodies may involve an additional challenge for mission designers.
- Thermal stress, including extremely wide temperature cycles which can require the electronics to survive to temperatures from -55ºC to +125ºC or beyond in a repetitive manner.
- Radiation-rich environment, including charged particles such as protons, electrons or heavy ions, but also neutrons, X-rays or Gamma rays produced inside or outside the solar system. Radiation may degrade the performance of materials used to build spacecraft, including electronic devices.
- Geometry limitations. Most satellites are currently launched in a folded configuration due to volume limitations imposed by launch vehicles. However, these satellites must be unfolded once in-orbit to reach their operating configuration. The unfolding process involves a high dependability challenge as an error can be catastrophic for the mission.
- Distance to Earth. Most missions are unavailable or extremely expensive to be repaired in orbit, and communications with missions going beyond Earth orbit may be affected by delays associated to the distance. For that reason, autonomous behaviors are commonly built in space missions, especially to deal with situations that need to be tackled in a short period of time, such as emerging problems but also programmed behaviors such as landing on another planet.

2.2. Radiation effects on electronics

Among others, one of the most distinguishing challenges faced by space missions is the exposure to radiation. As a consequence, it has been addressed by the space industry since the space race on the 1950s. From that point in space history, the nature of radiation phenomena and its consequences to electronic devices have been continuously investigated [17], [18]. The growing understanding of radiation threats has been a permanent driver to develop increasingly dependable, ambitious and successful space missions.

Radiation presents many risks for spacecraft, we now focus on the radiation effects that impact the most on electronics reliability in space.

2.2.1. The space environment

The most relevant radiations concerning electronic devices and circuits in space are photons, electrons, protons, neutrons and heavy ions. The origin of these radiations are energetic episodes or locations occurring in the universe, such as stars, supernova explosions, star collisions or even the Big Bang [18]. The natural abundance of diverse radiation types is different as they are produced in different quantities by different radiation sources. Moreover, radiation is not equally spread across the universe, as it is affected by the proximity to the radiation source and also to other factors such as the presence of magnetic fields or other radiation sources, so the radiation context is dependent on the location and the moment of exposure to space.

Space environment is defined as "the conditions of the space radiation for a given location or orbit" [18].

The radiation environment in the solar system is strongly influenced by the sun, which acts at the same time as a source and a modulator of radiation. The sun is a source of electrons, protons and heavy ions. Two different processes are responsible for radiation emission from the sun: electromagnetic emissions (irradiance and solar flares) and mass emissions (solar wind and coronal mass ejections). Coronal mass ejections and solar flares are the most hazardous events for electronics in space as they produce high fluxes of energetic particles (around $10^6 particles/cm^2 s$ with $> 10 MeV$ energy). The solar activity has a cyclic intensity, repeating every 11 years consisting in 7 years of solar maximum, when emission processes are more frequent and more intense, followed by 4 years of solar minimum, in which the number and the intensity of such processes is lower [18]. The cyclic variability in solar activity acts as a modulator for the radiation environment in the solar system as discussed later.

Nevertheless, not all the radiation in the solar system comes from the sun, as radiation coming from more distant sources can also reach the solar system. High-energy charged particles originated at diverse sources outside the solar system are often referred to as galactic cosmic rays (GCRs). GCRs are composed by 90% ionized hydrogen (protons),

9% ionized hellium (alpha particles) and 1% heavier ions, including all naturally occurring elements, but the abundance of each type of ion is different and drops significantly for atoms heavier than iron. Heavy ions produced at extreme energetic events in the universe can reach the solar system with energies up to $10^{20} eV$, much higher than previously described solar particles. However, the flux of GCRs in the solar system is just about $1 - 10 particles/cm^2 s$, wich is much lower than solar particles flux. Thus, GCRs can be much more energetic, but are much less common than solar particles. Despite their low flux, ultra highly energetic particles represent a hazard for electronics in space as a single impact may trigger an error. The sun cyclic activity modulates the flux of GCRs in the solar system in an anticorrelated way: at solar maximum periods, the GCR flux is reduced by almost 20% respect to GCR flux levels at solar minimum periods [18].

In addition to the sun and GCRs as the two main radiation sources in the solar system, an additional phenomenon must be considered as it is highly relevant in the radiation environment surrounding planet Earth: the charged particles trapped by Earth's magnetosphere. The geomagnetic field on Earth can be approximated as a dipole that interacts with moving charged particles resulting in a perpendicular Lorentz force which is proportional to the particle velocity and field strength. As a result, Earth's magnetic field prevents charged particles to enter Earth's atmosphere by repelling, diverting or capturing them. Once captured, particles cannot escape and conform a set of toroidal regions around the Earth known as Van Allen belts. Two Van Allen belts are typically considered: the inner Van Allen belt and the outer Van Allen belt; both are thicker at the equator, where the magnetic field is more intense. The inner belt goes in the range of 1200-6000 km of altitude and is mainly composed of electrons and protons. The outer belt is located in the range of 13000-60000 km of altitude ans is mainly composed of electrons. Geomagnetic field of Earth is not perfectly centered nor aligned with Earth rotation axis, so the shape of radiation belts is not perfectly symmetrical, but has irregularities. The most relevant is the South Atlantic anomaly (SAA), that produces an increment on the flux for electrons and protons at low altitudes over the South Atlantic ocean and South America. The high flux and energy of particles in the radiation belts are a hazard to electronics experiencing such conditions as the the probability of receiving a particle strike on a critical part of the circuit increases and the amount of radiation dose accumulated may also lead to failure. Solar activity modulates the radiation present at the radiation belts as it interacts with both affecting their composition (inner belt losses protons at solar maximum periods) and shape (belts get compressed when the solar wind is more intense; outer belts gets thicker at solar maximum periods) [18]. Trapped particles belts are naturally produced by the magnetic field of planets; in the solar system, not only planet Earth, but also other planets present trapped radiation environments around them, being Jupiter the most intense in the solar system.

Finally, not every particle arriving the vicinity of Earth is prevented to enter Earth's atmosphere by the magnetic field. As a consequence, some space radiation also reaches ground. When entering the atmosphere, charged particles interact with atoms present

on it, producing secondary particles that trigger a chain of nuclear reactions which is commonly known as shower of particles [19]. Typically, from a single original particle entering Earth's atmosphere, many nuclear reactions occur, resulting in a continuous flux of elementary particles. In the range between ground to 20km altitude, the atmospheric flux is composed mainly by neutrons (also protons, electrons, muons and pions in minor quantities). Atmospheric radiation can affect electronics on Earth and is especially relevant for air transport dependability; atmospheric flux at typical commercial flight altitude is in the order of $10 particles/cm^2 s$. At sea level, atmospheric flux decreases to about $0.1 particles/cm^2 s$, which is much lower, but still can affect electronics operating on ground. In addition, neutron atmospheric flux is more intense at the poles, where the geomagnetic protection is less effective, than at the equator [20].

Effects produced by radiation are different according to the nature and the abundance the radiation in each case and mission duration. For spacecraft, short missions on low Earth orbit (LEO) are the most benign since they stay below Van Allen belts, reducing experienced irradiation from trapped particles and being protected from most particles by magnetosphere at the same time. Missions going further must take into consideration the adverse environment described in this section. Along the years of space exploration, models have been developed to represent the harsh conditions of outer space. Models help to define the radiation dependability requirements for spacecraft by knowing basic mission parameters as launch date, mission duration and orbit or trajectory [17]. The Space Environment Information System (SPENVIS) web interface [21], created by European Space Agency (ESA), provides access to available models to forecast the radiation experienced by a given mission.

Next subsections give an overview of the most common radiation effects considered in space electronics.

2.2.2. Cumulative effects

Exposure to radiation causes damage to electronic systems that can get accumulated over prolonged time, limiting system endurance [17]. Cumulative effects are driven by the total exposure to radiation, which is called the fluence [17]. Degradation is caused by the energy transferred from the radiation particles to the materials forming the system. If the energy is transferred by the particle interaction with electrons in the material atoms, electrons gain energy and electron-hole pairs are created along particle trajectory, this is called ionization. In contrast, if the particle interacts directly with the atomic nucleus of the material, it can transfer kinetic energy and modify its original position or even trigger a nuclear reaction. Since atomic nucleus represent a small portion of the atomic size, particle to nucleus interaction has low probability resulting that ionization is the dominating interaction process.

Two main cumulative damage effects can be differentiated depending on the interaction type: total ionizing dose (TID) and displacement damage dose (DDD).

2.2.2.a Total ionizing dose

Total ionizing dose is the amount of radiation energy transferred to a material through ionization. Ionization can occur for photons and charged particles (electrons, protons and heavier ions). In the space environment, charged particles, particularly electrons and protons, are the most relevant concerning TID effects. The required energy to create an electron-hole pair (E_p) depends on the material, being $E_p = 3.6eV$ for silicon and $E_p = 17eV$ for silicon dioxide, which is a very common insulator material in electronic devices. Radiation can create electron-hole pairs in materials as long as it has higher energy than E_p. Typically, the energy of a single particle may be orders of magnitude higher than E_p son it can create thousands of electron-hole pairs in the material along its path. Total ionizing dose can be measured by quantifying the amount of energy deposited in the material by ionization. The official unit is the Gray (Gy), which equals to 1J of energy deposited in 1kg of matter. However, in the space community is more common to use the rad (for radiation absorbed dose) unit to express TID. Conversion between Gy and rad is given by $1Gy = 1J/kg = 100rad$.

Total ionizing dose degrades electronic devices performance mainly by altering the properties on insulating materials. Upon electron-hole pairs creation, a portion of them will recombine producing a zero net effect. However, in silicon dioxide, electrons mobility is much higher than holes, so after ionization, some electrons will sweep out the oxide avoiding recombination. Recombination is highly dependent on the applied field to the oxide during ionization; if an electric field is present, it will pull electrons and holes apart, reducing the number of recombined holes. Non-recombined holes will remain relatively immobile in the oxide (oxide traps) causing a positive net charge that is proportional to the accumulated dose. External fields and temperature may lead holes to move through the oxide and reach the interface with silicon, remaining there (interface traps). In Metal-Oxide-Semiconductor (MOS) devices, positive charge on gate oxides produces a negative shift in voltage threshold that alters response times, reduces switching speed and increases leakage current. In an extreme situation, the threshold voltage shift may result that most common negative-channel MOS (NMOS) transistors cannot be turned off and positive-channel MOS (PMOS) cannot be turned on [22]. Due to MOS technologies miniaturization, gate oxides are now much narrower than some years ago, so the impact of TID in gate oxides is now lower and resulting devices are more robust. However, oxides are not only present in gates, but also in other isolation structures, which size is larger than gate oxides such as silicon on insulator or shallow trench isolation technologies, that can also induce TID degradation. In bipolar technologies, passivation oxides can also become positively charged, degrading transistor gain and increasing base leakage. An additional effect is particularly relevant in bipolar transistors as total ionizing dose degradation is not only dependent on the dose but also on the dose rate, resulting in higher degradation when the exposed radiation dose rate is low: this is called enhanced low dose rate sensitivity (ELDRS) [23].

2.2.2.b Displacement damage dose

Displacement damage dose is produced by the accumulation of successive interactions between incident particles and the nucleus of the atoms in the material. This is the main interaction process in the case of neutrons, as they have neutral charge and cannot interact with the electrons of an atom, but also can be caused by electrons, protons and heavier ions. Upon an incident particle collides with the nucleus of a semiconductor in an integrated circuit, it may displace it from its position as long as it has enough energy. Interaction can occur by any of the following processes: Rutherford (i.e., Coulomb) scattering, nuclear elastic scattering and nuclear inelastic scattering. Upon first displacement, the displaced atom may have enough energy to provoke new interactions and the original particle may produce them as well. The simplest DDD damage mechanism is the creation of vacancies (empty places in the crystalline lattice) and interstitials (atoms that are placed in a non-lattice position). However, collisions may also provoke nuclear reactions that originates new atoms, ions and neutrons as subproducts, degrading the quality of the material. Defects may be created far apart one from another, which are known as isolated or point defects, but also may be concentrated in localized regions with many defects, which are called clusters. The interaction process and the provoked defects depend on the original particle type and energy. Typically, isolated defects are created by particles with lower energies while highly energetic particles usually creates both isolated and cluster defects.

Displacement damage dose degrades electronic devices mainly by creating defects on the crystalline structure. Nevertheless, DDD effects occur in very low proportion compared to TID effects so the damage is only relevant in those devices for which low defect count may lead to erroneous behavior. It is typically considered that bipolar technologies, solar cells and optoelectronic devices are the most sensitive to DDD efects [24], while they are considered irrelevant in MOS technologies. However the continuous miniaturization in MOS technologies can produce that defects produced by DDD may be large enough to provoke faults.

2.2.3. Single-event effects

As described in previous section, ionizing radiation mostly produces electron-hole pairs in materials by ionization process. When the material is an insulator, then charge trapping is produced, leading to TID effects described above. However, electron-hole pairs can also be generated in conductor or semiconductor materials, including silicon. In this case "transient effects can be produced from individual particles that can disrupt system operation" [17], known as single-event effects or SEEs. Single-event effects are not time dependent, as they can occur at any moment since just one single particle hit may trigger them. The driving parameter considering SEEs is the amount of particle hits, with enough energy, per time unit, which is called the flux [17]. The higher rate of particle hits, the higher the probability of experiencing SEEs by one of them. Ionization process can

be produced directly by an incident particle across the material or by indirect ionization produced by secondary particles generated after nuclear interaction between the incident particle and the atoms of the material [25].

Ionizing particles deposit energy in the semiconductor material along the path by creating electron-hole pairs. Particles may be able to get through the material or they may be stopped and remain inside the device structure, attending to energy considerations. Electron-hole pairs are generated at expense of energy transfer from the incident particle to the material, until the particle energy gets below the minimum energy to produce more electron-hole pairs ($E_p = 3.6eV$ for silicon), when it stops. This process depends on the initial energy of the particle, the length of the path along the material and the ratio of energy transferred by length unit, which is known as linear energy transfer, or LET. The total distance travelled by the particle inside the material is called the range. Linear energy transfer value is not constant along the range, but it increases along the path until reaching the Bragg Peak, when the LET is maximum and suddenly drops just before the particle is stopped by the material. LET is dependent on ion mass and energy; different ions with different energies can have the same LET. Linear energy transfer is given in $MeVcm^2/mg$, which equals to energy deposition per path length in MeV/cm normalized to material density in mg/cm^3 [25].

Not all particle strikes may produce observable single-event effects in integrated circuits, even if the energy deposited is high. SEE occurrence is also affected by the part of the circuit being hit, the angular orientation of the path with respect to the device and the existing field on the vicinity of the track, among others. It is considered that a SEE may be produced if the particle "strikes a sensitive region of the microelectronic circuit" [25], that is a part of the circuit in which the sudden appearance of free electron-hole pairs may lead to an error. Extensive research have been carried out since the 1960s to model, characterize, predict and prevent the occurrence of single-event effects in integrated circuits [25].

Single-event effects can be categorized attending to the affected region of the device and also to their consequences on its operation. Destructive and non-destructive SEEs are typically considered.

2.2.3.a Destructive single-event effects

When a particle hits an electronic circuit, it may “result in permanent degradation or even destruction of the device, provoking so-called hard errors" [26]. Four types of hard errors due to SEEs are the most common, namely single-event latchup (SEL), single-event burnout (SEB), single-event gate rupture (SEGR) and single-event snap-back (SES) [26].

- Single-event latchup (SEL) is a potentially catastrophic effect that leads an electronic device into a sustained state of high current consumption that, if not reverted, can lead to device destruction by thermal runaway. Such high-current state is called latchup and is characterized by a low resistance path between the power supply

and the ground of the device that remains, latches, after the triggering event is removed. Latchup condition may not only be triggered by radiation, but also by electric or electromagnetic interactions. However, radiation-induced latchup (SEL) is an issue since devices that do not exhibit latchup in normal operating conditions may experience latchup due to radiation.

- Single-event burnout (SEB) is a potentially destructive effect that affects power bipolar transistors and N-channel power MOSFETs by producing an internal excessive current. The electron-hole pairs generated by the incident particle may trigger a bipolar parasitic structure that produces avalanche multiplication, resulting in a high current state that, if not limited, may damage the device by self-heating leading eventually to thermal runaway, or burnout.

- Single-event gate rupture (SEGR) is a destructive effect that leads the gate dielectric of power electronics MOSFETs to break under an excessive electric field. Electron-hole pairs produced in the body of the device may diffuse under a moderate electric field in such a way that the holes are accumulated under the gate dielectric, provoking a transient increment of the electric field in that region. Even if the power device is operating in a system according to specification limits, the electric field increment due to radiation can provoke that the dielectric is locally exposed to an excessive field that leads to dielectric rupture.

- Single-event snap-back (SES) is a "stable regenerative condition similar to latchup" [26], causing an excessive current in NMOS transistors that can permanently damage or even destroy the device. Electron-hole pairs created by radiation near the drain junction of the NMOS transistor may trigger avalanche multiplication in a parasitic bipolar structure. For very energetic radiation strikes, avalanche condition may persist in time long enough to become self sustained and producing a high current state that can damage the device. An additional necessary condition for sustained snap-back is that an external circuit must drive enough current to the device, which is typical for I/O circuits. Single-event snap-back can be removed by turning on the NMOS transitor to reduce the drain voltage, or by removing power to the device. SES is not produced in PMOS devices as the avalanche multiplication is much lower for holes than for electrons.

2.2.3.b Non-destructive single-event effects

When ionization effects produced by the impact of a particle provoke a malfunction with no permanent damage on an integrated circuit, then it is called a soft error. Free electron-hole pairs generated by an ionizing particle strike on an integrated circuit may be effectively collected by the presence of an electric field, resulting in a transient current pulse that alters device operation but does not produce direct damage. In semiconductor integrated circuits, most sensitive regions to this effect are the reverse-biased junctions.

This is for example the case of the drain area in a bulk NMOS transistor, which may be reverse biased to act as an open switch. In the event of a particle hit, the charge collected by the electric field may appear as a short current spike at the drain during a process called charge collection [25].

The transient current pulse generated by a particle striking an integrated circuit can become a soft error if it alters device operation. Four types of soft errors are commonly considered according to the way they affect the behavior of the device, namely single-event transient (SET), single-event upset (SEU), single-event multiple-cell upset (MCU) and single-event functional interrupt (SEFI) [25], [27].

- Single-event transient (SET) is an illegitimate signal value produced by radiation transmitted throughout an integrated circuit. It is typically considered when the particle strikes a transistor which is part of a combinational logic path, producing an erroneous change on a signal value. The current pulse provoked by the particle may result in a voltage transient that mimics a legitimate signal on the device, and thus it may be propagated throughout the circuit and be stored on one on more registers. For that propagation to occur, the pulse must be wide enough to survive to technology propagation delays and parasitic capacitance. As the SET pulse propagates through the combinational circuit, its shape can be modified by gate transition times and capacitive effects. When reaching a storage element, the pulse must be wide enough and arrive at the correct time to be eventually captured. As newer devices are scaling down sizes and increasing clock speeds, they are more prone to experience SET effects. A fault in an integrated circuit due to an SET may or may not provoke an error, for example, if the erroneous signal is masked before being stored. Masking may be produced due to electrical, logical or timing factors related to device operation.
- Single-event upset (SEU) is the faulty alteration of one bit of stored information produced by a particle strike. In this case, the particle may strike a transistor that is part of a storing element in a circuit, such as a flip-flop or a memory cell. If the charge collected by the circuit is enough, the stored value may flip from logic '1' to '0' or vice-versa, which is called a bit-flip. A fault in an integrated circuit due to a SEU may or may not provoke an error as if, for example, the wrong stored data is overwritten with correct data before affecting device behavior.
- Single-event multiple-cell upset (MCU) is produced when a single particle hit produces more than one faulty bit in an device. Device miniaturization produces that transistors are built very close one from another, so the charge deposited by a single particle may be shared between two or more storage elements, provoking the alteration of more than one bit. MCUs are more likely in modern technologies with reduced transistor sizes and higher densities, and become an increasing concern because error mitigation techniques are usually designed to cope with single errors (SEUs). The energy and orientation of the trajectory of the particle with respect to

the electronic device also affects the probability of experiencing MCUs. A particular case of MCU is the single-event multiple-bit upset (MBU), where multiple bits in the same memory word are affected by a single particle. Faults due to MCUs, as SEUs, may or may not provoke errors in integrated circuits as if, for example, the corrupted data is never used by the device.

- Single-event functional interrupt (SEFI) occurs when the particle strike produces a device to lose functionality. It is an error mode that mainly affects complex devices in which the affected region is intended to control device operation, and the alteration can turn the device into an unexpected or forbidden operational mode. SEFI condition can be typically recovered by system reconfiguration, resetting or power cycling the device. Unlike previous effects, that may or may not provoke a device malfunction, SEFI conditions are a serious problem in complex circuits such as on-board microprocessors that control other parts of the system, because may remain inoperative for a period of time, which can be crucial for the correct development of the mission.

Despite soft errors do not directly trigger a technology-intrinsic potentially destructive mechanism in the device, as hard errors do, the behavior of an affected device may be altered as a result of ionization and the consequences of this alteration may be catastrophic depending on the overall system design. Increasing miniaturization of devices bring higher incidence of SEEs and soft errors, introducing challenges that must be overcome to fulfill dependability needs of current and further developments [28]. "Left unchallenged, soft errors have the potential for inducing the highest failure rate of all other reliability mechanisms combined" [29].

2.3. Addressing radiation effects

Sensitivity of a system to radiation effects may determine its suitability for a specific space mission. Performance of a device or a system under radiation is affected by its constituent materials, device design, fabrication and packaging processes and, if applicable, system implementation and integration. Electronics technology has evolved to overcome the radiation threats of space environment and develop devices and systems that can survive and perform properly in such extreme conditions. Several tests are conducted to prove that they will perform as expected under radiation before actually being considered to be sent to space.

2.3.1. Radiation hardening

Introducing specific actions to improve dependability of electronic systems under radiation is called radiation hardening. A wide range of techniques to achieve radiation hardening have been developed since the beginning of the space industry, and they can

be applied at various development levels. Devices implementing radiation hardening techniques are called radiation hardened, or "rad-hard" devices.

Radiation Hardening By Process (RHBP) [30], [31] is considered when the hardening is obtained by optimizing the device fabrication process to minimize negative radiation effects. RHBP includes techniques such as device doping modifications, insulator optimizations and the use of specialized materials. As an example, transistor isolation trough the buildup of oxide structures such as Shallow Trench Isolation (STI) and/or buried oxides (BOX) in Silicon on Insulator (SoI) technology can reduce SEE sensitivity. However, by adding more insulation structures, the device may be more affected by TID. To minimize that, special materials, such as high-k materials, are used at the insulators, and they are doped according to specific profiles. Other techniques such as triple- or quadruple-well structures and epitaxial substrates can be used to isolate transistors and reduce SEE sensitivity.

However, the production of RHBP devices is a very small portion of the global electronics market and so it has two major associated drawbacks: the cost and the performance. Very few RHBP manufacturing sites, or rad-hard foundries, are available around the globe, and the investment on the development of a radiation hardened fabrication process is high, which result in a high cost per device. RHBP also means that, once a fabrication process has been first developed for commercial applications, it must be then modified to comply with radiation requirements. The modification itself can take years and the introduced changes may negatively affect device size, speed or power consumption in comparison to non-hardened devices. As a result, when a new RHBP technology becomes available its performance typically lags behind two or three generations compared to the state of the art commercial technologies. Nevertheless, RHBP effectiveness is limited and devices manufactured according to RHBP standards are still sensitive to radiation effects. The aim to achieve very high figures of dependability and performance require to identify the most efficient manner to apply radiation hardening not only at device fabrication, but also at higher hierarchical levels [15].

Radiation Hardening By Design (RHBD) [15], [31] is the term given to the hardening achieved by introducing modifications on the circuit topology, layout or system architecture. Unlike RHBP, which only affects device fabrication, RHBD can be applied to a wider range of hierarchical levels such as the transistor level, the component level and the system level. By using innovative design and layout methods, RHBD can effectively mitigate radiation effects on components manufactured at commercial microelectronic foundries using completely standard commercial processes, although RHBD techniques can also be used at rad-hard foundries.

RHBD techniques are not aimed to reduce nor eliminate TID-induced degradation or SEE appearance, but they mitigate their effects instead. At component design, transistors or larger cells can be redesigned to overcome radiation effects. As an example, transistor geometry can be modified to avoid undesirable TID effects as radiation-induced edge leakage. If a transistor is built in a way that the gate completely surrounds the source (or

the drain) inside it, the resulting transistor has no edges and thus, is not affected by TID induced edge leakage. That type of transistor may present some drawbacks compared to an equivalent transistor with standard linear geometry, such as increased area and increased parasitic capacitance resulting in higher power consumption and higher delays. Nevertheless, those penalties can be equivalent to approximately one generation, instead of the two to three generation penalty of RHBP techniques. Other RHBD techniques to mitigate TID effects include the use of enclosed source transistors or doped diffusion rings.

RHBD techniques can also be applied to mitigate SEE effects at transistor level by, for example, increasing the width of the transistors that, with higher node capacitance and drive current, may reduce its sensitivity to SET and SEU although they may increase power consumption. At circuit level, introducing excess capacitors and resistors in data lines may mitigate SET and SEU by filtering short pulses, although they may increase propagation delay and reduce operation frequency of the circuit. At cell level, the dual interlocked storage cell (DICE) [32] introduce spatial redundancy in a more efficient manner, compared to other redundancy forms, by reducing SEU incidence with just 2x area penalty. Such solutions increase power consumption and area as the hardened cells require more transistors to be implemented than the unhardened ones. At system level, information redundancy can provide error detection and correction (EDAC) and scrubbing capabilities in memories to mask SEUs and cell interleaving can be applied at layout to minimize MBUs. An additional technique to mitigate SETs at circuit level is to artificially introduce controlled propagation delays on lines. At circuit level, SETs can be mitigated by avoiding long combinational chains that may induce pulse broadening. At system level, SETs can be mitigated by increasing the device power supply voltage, although it may increase sensitivity to SEL. Regarding SEL mitigation, it can be efficiently achieved by using RHBP techniques, such as SoI, but it can also be accomplished using RHBD at layout level by adding geometrical distance between well edges and active regions and at system level by lowering the power supply voltage. These techniques have drawbacks such as area increments and lower device operating speeds, respectively.

Finally, at device level, the packaging of the device also plays a role in shielding the sensitive parts of the device from the effects of radiation. In some cases, some components are covered with layers of protective materials, which is called spot shielding. Shielding is a commonly used way to enhance electronic devices behavior against radiation, especially for low-energy particles. However, the interaction of radiation with shielding materials may also create undesired secondary radiation products, becoming a new source of radiation.

Redundancy is a RHBD technique that deserves special attention because of its effectiveness and versatility. Redundancy can be defined as the "addition of information, resources, or time beyond what is needed for normal system operation" [9]. Redundancy can improve the dependability of a system not only against radiation, as introduced previously, but also against other types of faults. Any form of redundancy is needed to implement fault detection and fault tolerance capabilities, which are desirable attributes of dependable systems. However, the introduction of extra resources may inevitably carry

important penalties in terms of size, weight and power consumption (SWaP), performance or others. The redundancy concepts and applications are addressed in this document regarding radiation hardening purposes. The concept of redundancy can be applied in different ways:

- Hardware redundancy, introduces physical replication of the circuit to increase dependability. It is the most common redundancy form in electronic devices and systems fostered by the miniaturization and cost reduction in semiconductors. Hardware redundancy can be implemented using passive, active or hybrid approaches [9].
  - Passive hardware techniques rely upon the concept of fault masking through replication and voting mechanisms. With this approach faults are prevented to become errors without any action performed by the system operator, resulting in an inherent fault tolerant system. The typical implementation of passive redundancy is triple modular redundancy (TMR) which basically triplicates the hardware and performs majority voting on the triplicated results to determine the correct output. If one of the three replicas has a fault, the other two will still be correct and the faulty result will be discarded by the voter. The voter is the major difficulty to implement TMR as if the voter fails, then the output may be incorrect and the whole redundant system would fail. Any component that, experiencing a fault, may lead to a whole system failure is called a single point of failure and great efforts must be put on its design to minimize error probability. The triple modular redundancy approach can mask one fault, and it can be generalized to extend its capabilities as N-modular redundancy (NMR). Being N odd, 2 faults can be masked using 5 replicas and a voter; 3 faults can be masked with 7 replicas, and so on.
  - Active hardware techniques are designed to detect and remove faults from the redundant system before provoking undesired errors. Active approaches, however, do not mask faults so erroneous results may temporally appear at system output. Thus such errors must be acceptable on the system as long they can be removed. Instead of fault masking, faults are detected and corrected by reconfiguration process, providing fault detection, location and recovery mechanisms. The most basic form of fault detection is duplication with comparison scheme. In this case, two hardware replicas are used and the result of each replica is compared. If the comparison reveals that the replicated results mismatch, then a fault has occurred within any of the replicas. The comparator is again a single point of failure as, if it fails, correct results may be interpreted as errors or, even worse, faulty results may be considered valid. Fault location accuracy would be given by the granularity of the replication and comparison implementation, but it is not possible to identify the faulty replica. In the case that any of the hardware replicas has a permanent fault,

then a system failure would occur unless additional redundancy is provided. To overcome this situation, standby replacement, or cold sparing, can be applied by providing additional replicas that are not used nor powered until needed. In the event of a fault, standby replicas can replace faulty ones, restoring the capabilities of the system. Cold sparing can have similar area overhead as NMR but it presents less power overhead as not all the replicas are active at the same time, although it consumes time for switching between replicas.

  - Hybrid hardware redundancy techniques combine the attractive features of passive and active hardware techniques. The basic approach is to combine the concepts of N-modular redundancy (NMR) and cold sparing. The objective is to achieve a fault masking system and, at the same time, to be able to identify faulty replicas on the redundant system and replace them with spares to extend overall dependability with lower overheads than using just a passive or active approach.

- Information redundancy, is the addition of extra information to data to enable fault detection, fault masking and ultimately fault tolerance. Common implementations of information redundancy are error detecting and error correcting codes, in which redundant information is added to data values, or even data values are redefined with new representations that contain redundant information. The basic approach is the parity code, in which one extra bit is added to a binary value to always produce an even number of ones in the case of even parity, or odd if odd parity is used. In the case that a single error exist in the binary value, the parity will change and the error would be detected. However, two drawbacks arise: the first is that it is not possible to recover the correct value and the second is that multiple-bit errors may be left undetected. Parity codes approach can be generalized to cover a wider range of faults in the form of Hamming codes, which adds redundant information to perform error detection and correction (EDAC) in an efficient manner. Typical implementation of Hamming codes is single error correction, double error detection (SECDED) approach which, if combined with scrubbing and interleaving, can efficiently reduce error sensitivity in electronic memories [15].

- Time redundancy, is used to overcome the limitations of previous redundancy approaches that need to add extra hardware to the system to achieve fault tolerance. As hardware is a physical entity, it produces size, weight, power and cost overheads that may be unacceptable for some compact, low power, low cost, applications. However, using extra time can be an inexpensive option for systems that experience idle periods. A basic form of time redundancy is duplication with re-computation, which mimics the duplication with comparison active hardware approach. In this case, the same operation is performed two consecutive times and then compared, if the results mismatch due to a single error, an additional computation is performed to identify the correct result. Time redundancy approaches, however, may have a single point of failure in the hardware that, if not replicated, may compute the replicas in

an equally wrong manner due to a permanent fault, giving identical wrong results that would be identified as valid. To overcome this limitation, specific checks can be programmed on the system to periodically check hardware correctness before performing critical computations.

- Software redundancy, can be applied in systems that integrate computers to delegate fault tolerance capability implementation on the software. The hardware overhead for software redundancy techniques can be minimal, but additional software and associated time overhead may be substantial, ranging from a few extra lines to an entirely replicated program. A basic software approach are consistency checks, that leverages some previous knowledge about the computed data to check whether is correct or not. For example, the magnitude of a value can be detected to be incorrect if it is outside an expected range. That approach can be also valid to check the correctness of memory access in computers. Another form of software redundancy are capability checks, in which different parts of the system are checked to verify correct functionality. For example, the software can check if different pieces of the hardware are working properly, as the memories or the ALU, or even the whole processor by performance measurements. Despite software does not break as hardware does, software faults can appear as a result of incorrect design, being a potential single point of failure. To overcome this limitation, several programs may be developed by different design teams under the same specification, which is called N-version programming. Even if each program version has faults, they can expected to be different as they have been committed by different designers. If N programs are available, they may be executed in a voting scheme, in a similar way as NMR hardware approach.

As seen above, RHBD paradigm goes beyond device level. At board level, protection circuitry can be added to prevent faults to propagate between components and even to mitigate SEL by introducing extra circuitry that power cycles the affected device. At mission level, spacecraft geometry affects the exposition of each component to radiation, and the structure of the spacecraft introduces shielding for all components inside it.

Nowadays, space missions demand high performance and dependability for space electronics producing that not a single hardening method can meet the desired requirements in radiation environments. Ultimately, a combination of RHBP and RHBD techniques are typically applied to electronics in space applications attending to desired performance, cost and availability considerations, and size, weight and power (SWaP) requirements.

By the combination of hardening techniques, it is possible to obtain highly dependable electronic systems, but such techniques also increment system complexity that may negatively impact dependability. Thus, to obtain dependable systems it is not needed to apply every existing hardening technique, but to select the more convenient ones attending to the application. Finally, testing is required to check whether the resulting system is compliant with dependability requirements for the mission.

2.3.2. Component testing

To determine and quantify the contribution of radiation hardening techniques to the performance of a device or system under radiation, it is commonly needed to evaluate it by recreating conditions as similar as possible to the ones present throughout the mission. However, it is not possible to recreate the very same space mission environment on Earth for several reasons. First, a space mission typically last for years, but it is unfeasible to perform such a long test on Earth before launching the spacecraft to space. In addition, vacuum and absence of gravity conditions are expensive to recreate. Finally, the radiation environment in space is too diverse, especially attending to particle nature and energy ranges, while on Earth it is only possible to recreate radiation conditions for a subset of particles and energies. Nevertheless, methods have been developed to obtain a representative prediction of the behavior of an electronic system in space by a combination of tests. As a result, a test may obtain whether a hardened system is dependable enough for the mission requirements and so the hardening process is complete. If the test results are deficient, the hardening process must be resumed or reoriented to enhance the radiation hardness of the system. The reason for testing is to increase the knowledge about components performance under specific radiation conditions and quantify their dependability, thus reducing the risk of failure of the mission [33].

To check the radiation hardness of a device, typically a standard method or guideline is followed. Several test standards are available depending on the type of device under study and the radiation effects to be evaluated. Standards are built based on available evidence and revised periodically. Standardization allows to design tests that can be repeatable and comparable by pointing the different aspects that must be considered. As a result, others may replicate similar test conditions and obtain similar results. Testing is usually performed on a single device but, in most cases, other components and devices are needed for the device under test (DUT) to operate properly. Because of that, a test setup or test fixture is commonly required to exercise the component providing at least clock and bias. To check for correct operation it is also needed to write device inputs and read from device outputs. The setup may consist on diverse components as power supplies, signal generators, FPGAs, oscilloscopes or even computers, that must be protected from radiation. It is recommended that the test is as much automated as possible to avoid human errors, as most tests are very repetitive and can last for long time. Automatic logging is also strongly recommended to record all test data, that will depend on the type of DUT and the type of test, for example input and output signals, power consumption, signal waveforms or timing of the test among others. After the test is finished, test results can be obtained from analyzing collected data, and reported according to standardized metrics [33].

Standardization of tests removes the need for designers to test all components used in a space system, but first they must research whether they have been previously tested and there is available data. Available data must be reviewed and can be leveraged as long as it suits the requirements of the target mission. By knowing mission parameters such

as the orbit and mission duration, the radiation environment can be forecasted [21] and then, radiation requirements may be established to meet the dependability target. Systemic analysis can also help to determine the most critical components in the design to prioritize testing. To obtain representative results, it is crucial to design a testing approach that represents as much as possible the targeted mission [33].

Electronic devices testing can be performed by directly exposing them to radiation, which is called radiation testing, or by recreating the faults by other methods, which is known as fault injection.

2.3.2.a Radiation testing and facilities

The most widely accepted method for radiation hardness testing and qualification of electronic devices is the validation in radiation testing facilities on Earth. There is a limited number of radiation testing facilities around the globe. Such facilities have sources of radiation available for research purposes, not only for testing electronic devices but also others like physics, chemistry, biology or medical applications. Attending to space industry needs, there is no facility that can fully mimic the space radiation environment, so the approach is to mimic the degradation induced by the radiation environment using different test types for each type of radiation effect. Total ionizing dose (TID) and single-event effects (SEE) are the most common types of testing, but also displacement damage dose (DDD) and prompt dose testing can be made [33]. Radiation facilities are usually very demanded for research or industrial applications so the cost and the availability are important drivers when selecting one for testing.

It is remarkable that it is not needed to test all components for all radiation effects. As discussed on previous section, some effects are intrinsic of a particular technology or family of devices. Hence, for example there is no need to test a CMOS device for ELDRS effects and DDD will only be tested for photodetectors, solar cells and other optoelectronic components [33].

The radiation source must be selected according to the objectives of the test and the characteristics of the device. As stated before, the radiation source ideally should closely match the expected mission environment. Despite the chosen radiation source, the dosimetry is a key of every radiation hardness assurance test. The dosimetry is the measurement of how much radiation the device under test (DUT) has received, and it is used to interpret the test results. Hence the accuracy of the dosimetry will directly impact on the accuracy of the test results [34].

2.3.2.a.1 Total ionizing dose testing

TID testing allows to know about the degradation of the electrical and functional parameters of a device while exposed to ionizing radiation until it eventually fails. In

space, TID effects are mainly produced by trapped protons and electrons. On Earth, there are laboratories with available sources of both protons and electrons, but the cost of such facilities is high. As an alternative, gamma and low-energy x-ray testing, that are much more cost-efficient methods to produce TID degradation, are used instead. The most common source of gamma rays used for TID testing of electronic devices is Cobalt-60 (Co-60) artificial radioactive isotope decay [34]. The source is commonly placed inside a radiation-shielded room and the electronics are placed in front of it to be exposed to radiation.

The main parameters of TID testing are the total dose absorbed by the device and the dose rate as well, which is how quickly the dose is absorbed [34]. Commonly available Co-60 radiation sources can provide dose rates which are orders of magnitude higher than typical dose rate at space environments. The goal is to characterize the several mission years TID degradation suffered by an electronic component in the minimum amount of test time. Available test standards used in the United States [35] and Europe [36] set the recommended dose rate between 50 and 300rad(Si) per second for accelerated, high-dose rate tests, to obtain representative results. The total dose accumulated by the device under test will be the product of the exposure time multiplied by the dose rate, that should be precisely given by dosimetry.

During TID tests, electrical measurements such as current consumption or leakage currents and operational checks must be performed on the DUTs. The guidelines allow the devices to be taken out of the irradiation chamber for a limited time of two hours to perform measurements before resuming the irradiation. Standards state that the devices should be all measured before starting the test, and also at least at five intermediate irradiation points before reaching the target dose for the test. Other non-radiated devices must also be measured at the same times that the radiated ones to serve as control. That way it is possible to notice the evolution of the degradation in the parameters and functionality. By measuring circuit parameters at several dose intervals it is possible to determine a safe level of accumulated dose at which the device will operate correctly.

Due to the cumulative nature of effects, performing TID testing is destructive for radiated devices that, even if they already operate correctly after the test, have accumulated so much dose that make them unusable in any mission. Because of that, TID testing is highly statistical and care must be taken at selecting the samples to be tested as they should be representative of the non-radiated parts. Data obtained from TID testing must be collected as data points and plotted as a function of total dose. Such plots shall reveal the evolution of parameters such as current, timing or functionality along the test [33].

The main objective of TID radiation testing is to determine how much dose a component can be exposed to before the accumulated degradation affects its dependability. Typically, components first experience parametric failure at moderate doses, observing that some of the electrical, timing or functional parameters degrade beyond catalog specifications but the components can still perform their function. Functional failure commonly happens at

higher doses, when the components partially or totally lose their functionality [33]. With this information, maximum levels of TID can be established for the device to operate correctly. When designing a mission, maximum TID levels can be contrasted with radiation environment models and mission length to reveal whether the component will fulfill mission needs.

2.3.2.a.2 Single-event effects testing

SEE testing is used to determine the sensitivity of electronic components to different types of SEEs when exposed to ionizing particles. In space, all types of SEEs are triggered by the ionization (direct or indirect) produced by individual incoming particles that generate electron-hole pairs in the active regions of the device. On Earth, several particle accelerators exist where electronic devices can be irradiated using protons or heavy ions, whose nature and energy depends on the specific facility. Particle accelerators can produce a beam of particles that is wide enough to irradiate the whole component at the same time, which is called broad beam testing. Broad beam testing also involves temporal and spacial randomness in the particles hitting the device, which is also representative of the space environment. Broad beam testing is a well established method to recreate space particle radiation conditions for electronic components evaluation [34].

The main parameters of broad beam SEE testing are particle species, energy, flux and fluence. Ion species, characterized by mass number, and energy, given in MeV, are directly related with particle LET and range inside device material, which are of high relevance when designing SEE tests and interpreting the results. Particle flux is the amount of particles per second and per square centimeter ($particles/cm^2s$) in the beam and affects the error rate observed during the test. In most facilities, the flux can be set to be constant and it is precisely measured by dosimetry. The fluence is the total amount of particles per square centimeter ($particles/cm^2$) , given by the flux integrated over a given time, or simpler the product of the flux and the test time if the flux is constant. Fluence determination is crucial to obtain accurate test results [33].

Commonly available broad beam facilities are composed by three radiation-shielded rooms. Two of them are for the particle accelerator and the beam line. A shutter is placed between the particle accelerator and the beam line to remove radiation from be beam line room and let access to place the DUTs in front of the beam. An additional room is provided for people to control the beam and the test setup. Heavy ion energies in space can reach several orders of magnitude above that the highest available energies at Earth facilities. Because of that, ion energy is not used as the parameter but the LET in the active volume of the device is used instead. For illustrative purposes, available facilities can provide ions with LET up to $95 MeVcm^2/mg$. To allow low range ions to penetrate deep enough to the active volume of the device, heavy ion testing is typically performed in vacuum to remove air interference. In addition, tested devices need to be de-packaged, de-lidded

and even thinned to reduce the amount of material that the particles have to travel through before reaching the active volume. Attending to proton testing, proton facilities available on Earth have energies up to 500MeV, which is enough to cover the entire range of proton energies in natural space environment. In the case of proton testing, the lower interaction of protons with air, package and device materials allow to perform tests without the need of vacuum and/or device de-packaging. In both cases, proton and heavy ion testing, the beams are highly directional so other devices put away from the beam direction will be protected from radiation effects. Because of that, it is possible to introduce equipment in the beam line room to perform measurements during the test. Beam tuning is needed for any broad beam facility to guarantee relevant test features such as beam uniformity and energy uniformity, which is that particles are equally spread throughout the whole beam spot and that all particles have a similar energy [33], [34].

Regardless of the type of SEE to be tested, all SEE tests share common characteristics. The most important feature is that SEE testing is not governed by test time as it is TID testing, but it is based on events. An event in this context are referred to the occurrence of an SEE. The test setup must account each event and register the type of associated SEE. The flux of particles in the beam must be adjusted to keep an affordable event rate. If the flux is too high, the event rate may be so fast that the setup is unable to record all the events. It is also possible that the events may accumulate and be accounted as a multiple errors, for example multiple SEU being accounted as an MBU. If the flux is too low, then the event rate may be too low to obtain a sufficient number of events, or the test time would become too long and increase the cost of the test. The number of collected events is important to provide confidence in results. If the number of events is too low, the uncertainty will be too high so the conclusions to be drawn from the results may be weak and/or wrong [33]. Uncertainty is related to the natural random distribution of particles in broad beam testing. If possible, a minimum number of 100 events is generally considered as a good reference as it is associated to a statistical uncertainty of 20% with a two-sigma (95%) confidence level established by convention [34]. The error intervals associated to each measurement obey a Poisson distribution and must be included when reporting radiation test results [37].

The main result of broad beam SEE testing is the measurement of the sensitive cross-section (σ) of a device against the evaluated SEE types. The cross section has units of cm^2 and is a representation of the total area of the device which is sensitive to a certain type of SEE for a given set of conditions. To obtain the cross-section value, the number of events are accounted for each type of SEE and divided by the particle fluence of the test. The cross section has to be calculated for each tested LET in the case of heavy ion testing or for each energy in the case of proton testing. In addition to changing the energy or ion species, there is an additional way to change the LET introduced in electronic devices under heavy ion testing, by modifying the beam incidence angle on the device [34].

Several test methods and guideline documents apply for SEE testing both in the US [38], [39] and in Europe [37]. However, they share most considerations and recommendations to

conduct SEE tests. All guidelines state that more than one piece of the same device must be tested, while keeping at least one non-radiated part as control. The sample selection must be representative of the non-tested parts, especially for destructive analysis, in which the behavior of the untested parts must be inferred from the test results. While not all SEE guidelines require that the component is actively used when the beam is on, such condition may affect the test results. For most devices, static testing may underestimate the device vulnerability to different SEEs as the parts of the circuit not being used may hide additional error mechanisms. To provide full error characterization, components should be tested in the same operating conditions that they are going to operate in the deployed mission [33], [34].

The main objective of broad beam SEE testing is to determine the cross section value as a function of LET, information that can be used to predict error rates in orbit. Typically components present low cross section values, or even no events at all, at low LET and then the cross section increases suddenly at a specific LET value known as the threshold LET. In the case that no errors are collected during a test, it is assumed that the next ion would cause the effect, so a minimum of one event must be considered. When the error count is zero, it can be due to the component being almost immune to a certain type of effect, but it must also be considered to review the dosimetry and the setup to discard any problem in the experiment. There is a threshold LET for each device and effect. For higher LET values, the cross section increases until reaching a saturation value. Cross section as a function of ion LET or proton energy is represented in log-linear plots by convention, and adjusted using the weibull fit. Device tilting is often used to increment the number of tested LET values and precisely determine the threshold LET value. From the fitted weibull curves, available tools allow to perform error-rate predictions for deployed missions by providing mission parameters such as the orbit or the solar cycle. Typical result from error rate prediction is the Soft Error Rate (SER), which is the amount of soft errors expected per time unit in a given mission [33].

Apart from heavy ion and proton testing, which are the most used broad beam tests, it is also possible to irradiate electronic devices with neutrons. Neutrons are of increasing interest since they are naturally present in the atmosphere and can provoke SEE by indirect ionization on avionic systems or even computer systems on Earth. Due to the low interaction of neutrons with matter, devices can be tested without modifying the package and even more than one device may be tested at the same time [33], [40].

2.3.2.b Fault injection

Fault injection refers to techniques that assess the hardness of electronic devices without exposing them to radiation. Several methods to introduce faults in electronic designs have been developed in the last years essentially to bring designers the ability to evaluate designs or hardening techniques in a more convenient and cost-effective way than using radiation sources. Especially, for any device implementing RHBD techniques, ensuring that they

are performing correctly is essential. When testing a device under radiation, the random nature in the produced effects may cause that some rare fault mechanisms are uncovered. A common advantage of fault injection methodologies above radiation testing is that the position and the instant of the fault being created inside the evaluated component or system can be precisely controlled, so it is possible to correlate it with the subsequent errors and analyze the underlying mechanisms. Additionally, most fault injection methods are much less expensive than radiation testing, allowing for exhaustive analysis that would become prohibitively costly if performed in radiation facilities. Fault injection has been adopted by companies and researchers to augment the capabilities of radiation testing methodologies [41].

An additional degree of freedom in fault injection is that the level of abstraction for injecting the fault can be selected from the transistor level to the subsystem level. Thus it is possible to evaluate the effects of faults within a device but also in complex systems with many devices to determine how the faults in one component reverberate in the dependability of the system. The test coverage is a remarkable aspect of fault injection and determines what percentage of the design is tested. For example it is possible to test only a portion of the device, or to test the whole device but only for a particular effect.

Two approaches are generally considered regarding fault injection: fault emulation and fault simulation [41]:

- Fault emulation methodologies rely on hardware-in-the-loop techniques to mimic the behavior of the fault inside the tested device. Sometimes the actual device under test is directly used, but sometimes a different hardware is used to emulate the first to ease fault injection process.
- Fault simulation methodologies require models of the system or device being tested and apply analytical methods to evaluate radiation hardness. Circuit modeling tools are commonly used for this purpose and test benches are created to exercise the model introduce the faults and check for resulting errors. An advantage of fault simulation is that the circuit can be abstracted so there is no need for the circuit to be completely designed to perform a simulation, bringing the ability to check the radiation hardness of the design sooner.

Fault injection approaches can be conducted using ad-hoc implementations to evaluate a particular effect on a particular part of the design. However, it is also possible to develop general purpose fault injection tools to systematically test designs for the most common types of faults. Despite the selected approach, fault injection results must be validated with radiation testing to ensure their accuracy. The similarity in the results of fault injection and radiation testing will be higher as the fault injection approach represents better the device under study. Fault injection can facilitate the path to radiation hardness, but radiation testing is mandatory to evaluate the dependability of any device under radiation before considering to use it in a space application.

Additionally to fault emulation and fault simulation, additional approaches exist for injecting faults by directly creating electron hole pairs in devices while controlling time and location parameters: focused ion beam testing and pulsed laser testing. Focused ion beam testing uses mechanical or magnetic methods to focus the beam from a particle accelerator in a specific location of the device. Also, the number of particles can be effectively controlled even to use one single particle for each injection. However, focused ion beam testing still requires for a particle accelerator facility, which are costly and have limited availability. Conversely, pulsed laser testing uses the light from a laser source to generate electron-hole pairs in the device in a controlled fashion. This approach is of special interest in the radiation effects community and is discussed below.

2.3.2.c Laser testing

In addition to broad beam testing, it is also possible to study SEE phenomena on electronic devices by the use of pulsed laser. Pulsed lasers inject light (photons) in the sensitive volume of the component that creates free carriers when the electrons in the valence band of the semiconductor absorb the photons and excite to the conduction band. This mechanism to generate electron-hole pairs is very different than the Coulomb interaction produced by particle strikes in radiation environment, however, it has been proven to accurately reproduce similar effects. Two approaches are used to produce SEE in electronic devices using laser, either using only one photon to produce electron excitation, called single photon absorption; or two photons, known as two photon absorption [42].

Laser testing is considered as a form of fault injection, also called laser fault injection. However, it has deserved much attention and become a complementary tool to broad beam testing which has huge acceptance among the radiation effects community.

The main advantages of laser fault injection is the possibility to inject faults one by one and to accurately determine the position and the moment that each fault was injected. This feature contrast with the random spatial and temporal nature of broad beam testing and provides a complementary tool to investigate the origins and mechanisms leading to SEE without radiation damage. In addition, the cost of laser facilities is commonly lower than radiation facilities, and their availability is usually higher, allowing SEE investigations in a more cost-effective way than particle accelerators. The only requisite for laser testing is that the active volume of the device can be optically accessed by the laser beam, so device de-packaging, de-lidding and even thinning and polishing may be needed.

Laser test equipment include automatic positioning of the laser beam in the three x-y-z dimensions, and also a microscope camera to actually see the shape of the device internal structures for reference. While the z dimension is used to adjust the focus and the depth of the injected light, the x and y dimensions can be used to position the laser in any point of the device surface. Laser tests can be designed in such a way that the laser injects faults in the whole device area or in predefined portions of the device to characterize the behavior of particular structures. The laser beam can be focused to a spot with diameter of $1\mu m$ or

less and the x-y steps can be set as low as the spot size to allow for an exhaustive scan. At each location, the response of the circuit is recorded with the injection position and, by correlating that with the device layout provided by the designer, the zones which are sensitive to different effects can be identified.

The capacity to precisely associate some portion of the device circuit to the occurrence of a specific SEE helps to investigate the mechanism under these effects and brings the opportunity to improve the designs to eliminate them. An additional feature of laser testing equipment is to synchronize the injection with the operation of the device, for example the clock, thus allowing the study of time-dependent effects. Moreover, the energy delivered by laser pulses can be much higher than the maximum LET for particle accelerators, replicating the effects of highly-ionizing cosmic rays for worst-case analysis. The pulse duration of lasers can be in the range of picoseconds, however it has been proved that its duration does not impact on the results of the tests as long as it is shorter than the response time of the circuit. Depending on the effect being evaluated, multiple pulses can be injected on each x-y location to obtain an averaged result. Additionally, some time may be needed for the injected fault to manifest as an error so it is recommended that the injection system waits a certain time between injections [41].

Pulsed laser testing has been proven to recreate almost all types of SEEs observed with heavy ions. The results from laser testing are generally used to identify SEE mechanisms and weak points in the circuit design to implement mitigation and improve the components design. Additionally, laser testing can be used to validate a SEE hardening approach that has been implemented after finding a weakness in a previous design.

2.3.3. Component qualification

Qualification testing is, according to NASA instruction EEE-INST-002, a set of "mechanical, electrical, and environmental inspections intended to verify that materials, design, performance, and long-term reliability of the part are consistent with the specification and intended application, and to assure that manufacturer processes are consistent from lot to lot" [43]. The qualification process can be considered destructive, as even the parts completing qualification may still be functional, they have probably been exposed to such extreme conditions that forbid their use in space. Because of that, devices shall be justifiably sampled to be representative of the non tested parts [43]. Most radiation testing standards include specific guidelines for qualification testing [36], [37] including the number of samples to be taken, the test conditions and how to control the sampling uncertainties.

Qualification of electronic components is part of radiation hardness assurance activities. Radiation hardness assurance (RHA) is a quality methodology that comprises the activities to evaluate whether a system operation will be correct in a given radiation environment during a given mission. RHA is a key part of a successful space mission by reducing the risk of radiation-induced failures in deployed systems. The gold standard to determine the

RHA of a system is through radiation testing, allowing to understand how radiation affects the design and also to determine whether the applied hardening to the device is enough to fulfill mission requirements [41].

Qualification activities are typically carried out by manufacturers of high reliability products and qualification results are included in product specification documents, along with the conditions under those results are valid. Qualified products are typically meant to be commercialized for general use in space. However, components may also be qualified by users when no radiation data is available or re-qualified if the available data is incomplete or not applicable to the target application. If providing non-qualified components, RHA activities correspond to the user.

2.4. Trends in space industry

The space industry has evolved since its beginning last century. Electronics technology innovations along with continuous research on radiation effects and hardening techniques have led to a sustained improvement in spacecraft performance and mission success. However, the increasingly affordable access to space has favoured new players to enter the space sector with different interests, questioning the established methods for spacecraft development, which are no longer the only way to mission achievement. The space industry is experiencing significant changes, that are discussed in this section.

2.4.1. Background: the so-called "traditional space"

The space exploration started in the context of the cold war in the 1950s and 1960s, driven by the competition to develop the associated technology, known as the space race. On those times, the ability to deliver spacecraft to orbit had the additional purpose of demonstrating intercontinental missile capability to the world. Due to size, weight and power (SWaP) constraints, the first spacecraft of human history were made up of the transistors that were just becoming commercially available, instead of most widely used vacuum tubes. The space methodologies were developed in a highly politicized and strategic scenario in which missions were supported entirely with government funding awarded to large commercial contractors. Missions were commonly developed for years and most of them consisted in a single or few large-scale spacecraft intended for scientific or strategic purposes. As a matter of national pride and for security reasons, risk balance against profitability was not a priority and missions were designed in such a way that failure was not an option and had to be avoided at any cost; this approach is known as risk avoidance [44].

To that end, every component in the mission, humans included, were rigorously tested under extreme conditions to ensure quality and reliability throughout the mission, increasing confidence and minimizing the risk of failure. Military and space assurance

protocols evolved at the same time as new missions were developed, fed by lessons learned from previous experiences and failure analysis. In general, they tended to pursue mission success by the use of conservative assumptions, which included the requirement for the use of radiation hardened (rad-hard) components whenever available, or rigorously screened military parts when not. Specifications regarding devices building materials and the traceability of lots and devices by strict manufacturing protocols were also considered. Additionally, parts exhibiting any destructive single-event effect were commonly prohibited. Years after, the guidelines were relaxed to allow the use of commercial parts just in case the required functionality and/or performance is not available neither in rad-hard nor military parts, as long as RHA could be assured by extensive lot-based testing and enough mitigation features were introduced. As a result, most missions developed under traditional space guidelines lasted largely beyond their initial mission length requirements, as for example the Voyager I and II, which have been operating for more than 40 years, and still they are.

The traditional space approach has demonstrated to conform a reliable methodology for more than 40 years, based on documented experience and requiring that the manufacturers provide access to exhaustive device data [45]. Traditional space methods are still used and valid in missions today, however, different trends have arisen, driven by a new context in space industry.

2.4.2. New Space trends

The global context in which the traditional space started has evolved through decades, becoming a very different situation in which governments and public agencies have much lower budgets available for space and science in general. A new movement, known as "New Space" was initiated at the end of the 1990s mostly by small research groups that produced small satellites barely following the traditional space methods. Such small satellites were focused as technological feasibility projects or educational platforms. Commercial-Off-The-Shelf (COTS) components were mainly used regardless of reliability or radiation effects, expecting high failure rates. The most representative trend for small satellite missions are the so-called CubeSat systems, that are based on a fixed form factor of $10 \times 10 \times 10cm$, which is called 1U. Larger satellites can be formed by stacking 1U cubes together known as 2U, 3U, etc. Small satellites have become very popular in the last decade mainly due to the reduced launch costs and especially the increasing capabilities of microelectronics that provide high performance functionalities with lower SWaP requirements than ever before [44].

New Space missions are usually associated with limited budgets and narrow development schedules, putting restrictions on the decision making, allowed lead times, and afforded test campaigns. In such scenario, failures may arise due to undetected development errors, but the risk may be justified by the associated reduction in cost and development time. The approach of risk avoidance at high costs from traditional space

is then converted into risk management [46] at adjusted costs, whereby the resulting system has to be good enough for the mission. Spacecraft failure may be an allowed mission parameter as long as system redundancy can be achieved by launching multiple identical spacecraft to form a constellation providing spare units, allowing that if any one fails, then another one can replace it. This New Space vision of mission profitability have attracted private companies to offer space-based products, such as communications services, beginning an era of space commercialization [44].

2.4.3. Use of COTS in space applications

Within this document, the term Commercial-Off-The-Shelf (COTS) is associated to assemblies or parts "designed for commercial applications for which the item manufacturer or vendor solely establishes and controls the specifications for performance, configuration, and reliability (including design, materials, processes, and testing) without additional requirements imposed by users and external organizations" [45]. COTS are available through manufacturer catalogs and are distributed "as is", without any additional testing after delivery. "The performance of COTS parts are guaranteed only against the contents of the vendor's datasheet and not to a customer controlled specification. In general, COTS parts are intended for the commercial marketplace and terrestrial, non-high reliability applications. For these reasons, special NASA and Military assurance requirements such as lot traceability, lot-based destructive qualification testing and radiation testing are usually not met" [47]. Using COTS devices in spacecraft, instead of rad-hard ones, may increase the risk of failure of the mission, as the radiation hardness is not assured by the manufacturer and probably has not been considered within device development.

However, COTS have been successfully used in space applications for decades [48], not only under the New Space paradigm but also in traditional space missions. Even the NASA developed a methodology [46] in 1998 to support the use of COTS components in its missions. Initially the reasons were to bring ultimate performance of newest devices to space, which were not yet available in rad-hard electronics. It is common that rad-hard devices lag two or more generations behind COTS counterparts, influenced by their respective market sizes. To use newest devices is part of the motivation for the use of COTS in New Space missions, but the main driver are budget and schedule constraints: rad-hard devices are commonly prohibitively expensive and present long lead times that New Space missions cannot deal with. In each case, both traditional and New Space paradigms, the parts selection must be made to fulfill mission requirements, especially mission duration and orbit [45]. If requirements accurately capture mission needs, then some risks may be assumed by selecting certain COTS devices with limited confidence levels, while others can be identified to lead to catastrophic consequences requiring high reliability rad-hard devices, or heavily tested and screened COTS components.

COTS should not be considered as an homogeneous set of devices, as they may be targeted to a wide range of different commercial applications each one demanding

particular requirements. The number of manufacturers and technologies associated to COTS devices are also diverse so it is important to deal with each component individually. The paradigm of commercial electronics seeks to maximize benefit by continuously improving manufacturing processes, including die revisions, process changes and/or device shrinking without notifying the user as long as the devices still comply with the datasheet. Because of that, it is common that even two parts of the same device may behave differently under radiation because they are physically different. Additionally, requirements and hazards must be evaluated for each component assuming that COTS and especially emerging technologies may be more vulnerable to radiation effects, or even exhibit new effects, than older devices. Existing guidelines about COTS usage in space claim to develop a risk analysis to identify the critical components, which are essential to the mission, and the ones that are not, for example status indicators, components with redundant backups or components that are only needed at the beginning of the mission [43], [46]. Determining how critical is each component in a system may help to manage the risk at device selection. However, when considering COTS there are commonly too many unknowns to perform an accurate quantitative risk analysis [46].

COTS parts must be used very carefully in space missions and, when radiation data is not available, radiation testing should be performed to gather as much knowledge as possible. Since access to detailed information about device architecture is generally limited, users have to generate their own information based on test data [45]. The challenge increases when considering complex devices such as microprocessors, which are commonly crucial components in spacecraft systems. Any limitations in understanding the architecture and failure modes of such components may lead to unpredictable behaviors that may be difficult or impossible to mitigate. "Although radiation testing is expensive, the cost is far lower than that of premature in-flight failures or last minute hardware changes" [46].

2.4.4. COTS testing and qualification

COTS offer great computational performance, functionality, cost, size, weight and power features. In addition COTS are already used for high reliability, critical applications on Earth like in the case of automotive sector. However, COTS manufacturers don't characterize parts against radiation, which is one of the largest challenges for space missions discussed previously. Considering the high effort and cost to send hardware into space, the mission success probability must be maximized.

Radiation hardness assurance methodologies seek that systems behave as expected once exposed to space radiation environment. This is also applicable to COTS, however, the RHA efforts are higly dependent on the mission [43]. The greater difference between COTS and rad-hard parts is that in the case of COTS the radiation testing is not performed by the component manufacturer, but has to be performed by the user. Considering the higher complexity of modern commercial components, integration levels and processing speed, testing commercial components becomes more challenging than testing rad-hard

devices, especially due to lack of information about component architecture. Commonly, COTS exhibit complex response to radiation, introducing issues in the RHA process [46]. COTS suitability has to be evaluated for each space mission instead of looking for a space grade COTS that could be used for any mission.

COTS have been successfully used in space missions, even for critical applications, by NASA when equivalent space or military grade parts are not available [45]. COTS selection, qualification, derating and screening were performed attending to the mission parameters and the purpose of the component, according to existing NASA guidelines [43]. However, radiation testing cannot bring quality in devices, but only determines if their quality satisfies the mission requirements [43].

Radiation testing introduces cost and time penalties in the development of a space system, reducing the benefits of using COTS parts [47]. Not having to wait for test results may reduce costs and time to market, as COTS parts are commonly affordable and readily available, however it may introduce unacceptable risks. Testing a COTS device to qualify it for a given space mission is called upgrading, and depending on the mission it may become more expensive and less effective than choosing an already qualified part if available [43]. An exception is the recurring use of a specific COTS component, for example the need for many identical devices in one mission, in different missions or a in set of satellites for a constellation. In such cases, the COTS qualification costs can be spread among all component units, resulting in a more cost effective solution than purchasing a high amount of rad-hard devices [44].

Nevertheless, qualification testing is not all what is required to obtain COTS components ready for space missions. As an statistical, and typically destructive process, qualification needs for providing an homogeneous sample of devices to obtain data from the tested ones to be representative of the untested portion. Unfortunately, the case of COTS introduces issues in the qualification process due to uncertainties and changes in the manufacturing process [43]. Die revisions or die shrinking may affect parts behavior under radiation due to changes in the microelectronic architecture that does not negatively affect datasheet parameters. It is common that manufacturers fabricate parts in more than one location around the world, each one using non identical processes. In addition, parts manufactured in one location may be packaged in other facility which may difficult the traceability of the devices, and manufacturers usually do not provide lot, wafer or facility information [44]. When radiation qualification testing and screening is performed to understand the risks associated to a COTS component, it is important to be sure that the device sample is as homogeneous as possible, to reduce uncertainties introduced by manufacturing variability [47].

COTS market size is huge, and many manufacturers and industries are involved so, as mentioned before, each component must be treated individually. Among all COTS a subset of them can be established attending to those providing additional data to support RHA activities, which NASA denominates COTS Plus: "COTS Plus is a part supported by

test data available to end users establishing random failure rate assumptions, performance consistent with the manufacturers data sheet and methods to exclude infant mortal parts, parts with latent defects, weak parts, or counterfeit parts" [45]. It is common that manufacturers offer a subset of their products targeted to high reliability applications such as medical or automotive ones by providing a controlled production process with good traceability at fabrication assembly and test, which may reduce uncertainties in qualification results, increasing the confidence on the upgrading process.

For COTS components which no associated radiation performance data, testing is highly recommended. To determine the suitability of a COTS component for a space mission, temperature cycling and radiation testing is performed, and performance characteristics are measured. The type of testing must be decided to estimate SEE rates, TID-sensitive parameters and overall circuit degradation. Parts that exhibit parameters outside datasheet specifications or excessive parameter drift are declared failing and, if an excessive number of parts from a given lot fail, the entire lot is discarded. Also, a life test must be passed by all flight candidate samples. Due to the high cost associated to SEE testing, it is usually first performed at proton beams since they are more accessible and only if the devices pass the test they are considered for heavy ion testing. Despite such considerations, testing applicability is limited since the circuit architecture of the device is unknown, introducing risks that must be managed. The common approach is to characterize the behavior of the device in a specific radiation environment, that must be representative of the targeted mission. [46]

In some cases, board level or box level testing is considered for missions willing to accept high risk. However, it can be difficult to replicate the accelerated failure factors that are usually considered for part level testing. Additionally, several parameters will be hard to measure and in the case that something fails it may be difficult to understand what happened [44].

When radiation testing is performed on a COTS component to qualify for a space mission, it can happen that the component does not pass the test and cannot be used in the flight system. In such a case, the cost increases: not only the supply of the device samples and the price of the radiation facility, but also the time spent and the need to search for another component to perform the required function. To mitigate that, several candidates are often selected and qualification process is performed on all at the same time. However, it could happen that all of them fail. Fortunately, there is an additional consideration to maximize success on COTS qualification, by attending to the known risks associated to each technology. Continuous innovation in CMOS technology brings smaller devices and lower power supply levels, which contribute to reduce radiation effects such as TID or SEL, though increasing SEU susceptibility. The use of a device manufactured in SoI technology would definitely provide SEL immunity, while using a bipolar device may introduce ELDRS susceptibility. The targeted commercial application for a given COTS device can also guide the selection process: while components for consumer electronics are usually optimized to perform well under the life cycle of personal devices under ambient

conditions, their operation under extreme temperature conditions may be quickly degraded. Conversely, a component targeted to industrial, medical or automotive equipment may resist adverse environmental conditions without failures. By carefully choosing the candidate devices, the success of COTS qualification process can be maximized. It is recommended to select components from intermediate levels of performance and reliability instead of the highest-performance ones, as long as are suitable for the application [44].

It is also possible to introduce mitigation at system level to reduce the risk associated to the use of COTS components in spacecraft like reducing the operating frequency and voltage to mitigate SEEs and controlling the operating temperature of the device to maximize device lifetime. Powering down devices when they are not used contributes to reduce TID degradation and the chance of experience destructive SEEs. Architecture level mitigation can be used to introduce redundant circuits in such a way that if a COTS fail in orbit, it could be replaced by another identical component. SEL protection circuits are also used for devices with risk of experiencing latchup [44].

2.5. Microprocessors under radiation

Microprocessors, as any other integrated circuits, may be susceptible to the basic radiation effects. Single-event effects (SEEs), such as single-event upsets (SEUs) and single-event transients (SETs) are of high relevance as they can trigger varied and often complex errors and failure modes [40].

2.5.1. Microprocessor errors

Errors in microprocessors can be classified as control-flow errors if they provoke an incorrect execution flow of instructions, or data errors if they only affect program data. Such categories are linked to the classic processor architecture divided in control path and data path, in which the data path performs arithmetic and logic operations while the control path executes the algorithm described in the program. However, there is no direct link between the fault location and the subsequent error, as data may be used in the program flow to take decisions and the control path is constantly loading and exercising the data computation units. For that reason, an error in the data path can end as a control flow error, if the wrong data was used to perform a branch instruction. Similarly, an error in the control path can result in a bad data result if the wrong value is loaded on the arithmetic and logic unit (ALU) or if the wrong operation is commanded. Additionally, modern microprocessors include additional features to the classic architecture that increase the complexity of the fault effects, such as multi-level pipeline, branch prediction unit, out of order execution capabilities, floating point unit or cache memories.

SEUs and SETs may result in varied failure modes depending on the affected microprocessor subsystem. Errors affecting the registers or the cache contents may corrupt

program data, but also the operation code of a program instruction or even a pointer. As a result, wrong computation results, known as silent data corruption (SDC) can be produced. However, more severe consequences may appear, such as program exception or even program crash may appear. In the case that the fault affects the control logic of the processor, instructions or data operations may be incorrectly performed despite the executed instruction and input data are correct, leading in most cases to an interrupt on the provided service (SEFI) [49].

2.5.2. Microprocessor testing

Testing microprocessor devices is particularly challenging as they can become very complex, by including a wide variety of resources as multiple input / output interfaces, several memory levels, more than one processor core and multiple processing modes. Additionally, the executed software, including the operating system, if present, determines the behavior of the hardware.

The main challenges for microprocessor testing, are recognized to be the "limited understanding of the internal architecture organization and the limited observability of the internal state" [33].

By analyzing the different processor resources, it is possible to identify sources for different types of errors, such as the cache or the core registers for SEUs. However, there can be memory or registers inside other resources such as the ALU. The combinational logic design of ALUs makes them prone to experience SETs. But there is more combinational logic inside a processor, such as the control logic of the pipeline or the out of order engine. This analysis can be helpful to determine the type of testing regarding which resources and effects must be tested.

2.5.2.a Microprocessor radiation testing

Test methods for microprocessors and SoCs have been proposed by NASA Jet Propulsion Laboratory (JPL) in [50] and [51]. According to them, the registers, the peripherals and the cache memories should be tested to understand the basic sensitivity of the device. Since some software is needed to test a microprocessor, it is also recommended to test the flight software if possible to replicate the flight conditions and understand how the basic sensitivity of the microprocessor affects the flight application. If not possible, it is recommended to test a representative software with the most similar workload as the flight application. This recommendation matches the classic approach "test as you fly and fly as you test".

It is not well understood how software affects the impact of SEEs on a microprocessor. When the flight software is not available or the testing is performed to generally characterize the response of a microprocessor under radiation, it can be difficult to obtain representative

results, as generally a software is needed to test a microprocessor. In fact, it is not straightforward to translate the individual cross section of microprocessor resources to the global cross section of a particular application, as only faults in active resources may affect execution. It is possible that some faults only impact in performance but do not produce wrong results, like in the case of a wrong branch prediction or an erroneous cache miss. Since the software does not generally use all the resources at the same time, the cross section of a given application is generally lower than the sum of the cross sections of each internal resource of a microprocessor. In addition, the order and timing in the use of the different resources in the application with respect to the fault instant may impact on the cross section as some faults can remain silent in unused resources and other can be masked or overwritten before becoming errors. The problem is that the use of some resources such as the cache or the registers by the processor is not completely understood [33].

Software benchmarks have been historically used to compare different microprocessor-based architectures or compiler technologies in terms of performance by the computer architecture community. When using benchmarks, two main features are pursued: (1) that the benchmarks accurately represent the operations and the workload of real applications, and (2) that the results can be compared between different organizations or research groups who are using the same benchmarks. A similar benchmarking approach may be highly beneficial when evaluating hardening techniques or error mitigation approaches in microprocessors working under radiation environments [52].

Common approaches intended to quantify the radiation hardness of circuit layouts or manufacturing processes may be used to evaluate the radiation hardness of any device, including microprocessors. However, when post-manufacturing mitigation techniques are introduced (such as RHBD techniques), it becomes necessary to quantify how these techniques impact on the overall system reliability, performance, size and power consumption, compared to the baseline version without mitigation. Unfortunately, test standards for microprocessors are still evolving and there is no available standard set of benchmarks for the evaluation of mitigation techniques. As a result, it becomes difficult to share or compare results between devices or organizations [52].

In [52], a set of software benchmarks for reliability evaluation, as well as a methodology for performing the tests, is proposed and evaluated. The main objective is to provide a set of different benchmarks representing different application workloads enabling the evaluation of different mitigation methods, which may be focused in a particular operation. Provided benchmakrs have a reduced footprint to enable implementation in a wide range of devices. Several benchmarks have been selected, focusing in realistic algorithms that may provide actual insight into the radiation effects on a spacecraft application. Some of the proposed benchmarks in [52] have become very commonly used in the microprocessor radiation effects community, such as Advanced Encryption Standard (AES), matrix multiplication, and Quicksort.

According to [52], when using software benchmarks to evaluate the radiation susceptibility of microprocessors, the reporting of the results must include reporting not only the reliability improvement between the mitigated and the unmitigated version of the benchmarks, but also the used algorithm, the input data sets, the compilation settings, the design tools and the runtime environment.

Applications run on baremetal implementation are directly executed on the processor hardware, however it is not uncommon to use operating systems (OS) in microprocessors for space applications. Operating systems are complex pieces of software that put an additional layer between the application and the processor resources, controlling scheduling and resource management. The impact of SEEs on the execution of an OS-based application may negatively affect the processor susceptibility to errors, even leading to extremely high error rates [50].

Complexity can also be increased in the hardware of a microprocessor, as in the case of the modern multicore SoC implementations, in which more than one processor instance is integrated in the same chip. While each core is capable of executing an independent application, in most cases there are shared resources between the processors such as the memories, peripherals or control logic that can provoke shared failure modes. Testing multicore SoCs is a challenge because the problem scales as more processor cores are tested as they must be simultaneously observed [33].

The most common approach to test microprocessor devices under radiation is to use evaluation systems rather than making custom boards or setups to reduce the effort and allow a cost-effective and reproducible evaluation of the component. Standard interfaces are typically used to input and output data from the processor during the test, but accessing to the internal processor state is a challenge. Boundary scan ports based on JTAG or other standards provide access to the pins and flip-flops of the device, however the implementation of such access is dependent on the manufacturer and may be difficult to use them for processor observation under radiation. Moreover, this approach is commonly asynchronous with the application, as it requires to stop the processor to gather all the required information, altering the execution timing and slowing down the test [33].

The analysis of microprocessor test results can be envisaged to obtain the cross section of a particular application or a given processor resource, depending on the software used during testing. Microprocessor errors are typically classified in silent data corruption (SDC) or detected unrecoverable errors (DUEs). SDC refers to erroneous data outcomes which have not been corrected nor detected by mitigation techniques and DUE refers to errors that, even being detected, cannot be corrected. In the case of multicore processor testing, it is important to identify the results of each processor to determine individual error rates and to pay attention on the error timing to identify shared failure modes that may occur in the same time frame [33].

2.5.2.b Microprocessor fault injection

Fault injection is commonly used to determine the vulnerability of microprocessors to faults in a cost-efficient way. However, the complexity of processing systems and executed applications makes almost impossible to cover the whole test space in terms of memory and register locations and clock cycles. For that reason, the test space can be reduced by only injecting faults in random locations and instants, which is called statistical fault injection (SFI), thus reducing the test time and computational effort. As a drawback, the accuracy of SFI results decreases as the test space is reduced.

A systematic approach commonly based on intensive fault injection campaigns intends to quantify the probability that a fault in a specific circuit location provokes a visible system error [41], which is known as the Architectural Vulnerability Factor (AVF). The AVF metric is commonly used to infer the response under radiation of a non-tested device by knowing the vulnerability of each module in the device and a normalized response under radiation of the manufacturing technology given by radiation testing on other devices manufactured with the same technology. In [53], a processing architecture is evaluated to obtain the error probability of different processor sub-modules under single bit upset fault model to orientate and optimize the hardening effort. In [54], a similar approach is extended to multi-bit upsets, revealing that the vulnerability of some circuit modules is considerably higher under such double- and triple-bit faults. To refine the correspondence between the estimated error rate given by AVF analysis and the real response of a device under radiation, a mathematical model is proposed in [55] to combine results from real proton and neutron broad beam radiation campaigns with simulated fault injection experiments, demonstrating high precision in the obtained model.

Diverse fault injection techniques can be proposed. As a general feature, techniques can only test faults in resources which can be accessed. One approach to inject faults in microprocessors is to use HDL descriptions of the circuit and instrument it to insert faults in the desired locations and moments, as for example the operation codes on the instruction memory or the variable or pointer values in the registers or the data memory [41], [56]. With this approach, the entire processor architecture is accessible to inject faults. However, HDL descriptions of processor cores are commonly unavailable, especially in hard-core processing systems. Another approach is to use a microprocessor software simulator for the fault injection, but the number of processors supporting this capability is also limited. In addition, software simulations are more time-consuming than hardware simulations and also the accuracy of fault injection results is quite limited because the model used for simulation may not be able to mimic the fault behavior of the actual processor [56]. The use of boundary scan, JTAG or similar debugging resources can also be used for fault injection [41]. In this case, the execution of the processor must be stopped to introduce the faults in the architecture, which makes this approach time-consuming. Additionally, the available resources for fault injection are limited to the ones accessible by the available interfaces.

Code Emulated Upset (CEU) [56] is a fault injection technique specially designed for microprocessors, by which faults are injected through the executed software. CEU approach consist in injecting bit-flips in the memory, the registers or any other software-accessible processor resource, during a simple software routine triggered by an interrupt. The processor must save its state before the interrupt and then execute the necessary code to introduce the fault in the desired location, and then return again to normal execution, when the effects of the injected fault can be observed. The main advantage of this approach is that it can be applied to any processor regardless of the complexity of the hardware or the application, with minimum impact on the circuit operation. As a limitation of this technique, faults can only be injected on the software accessible resources, leaving some other resources sensitive to faults untested. In addition, time resolution for fault injection is also limited to the execution of each instruction.

2.6. Microprocessor fault-tolerance

Microprocessors, as any integrated circuit, can benefit from generic purpose RHBP and RHBD techniques applied at transistor, device and system level to introduce fault avoidance and fault masking strategies. However, generic purpose radiation hardening techniques are generally applied systematically to the whole circuit and their effectiveness is limited as they are only intended to reduce basic radiation effects. In fact, microprocessors, as other complex integrated circuits, present a wide variety of internal subsystems and features that may lead to more complex error modes, especially when affected by SEE. Generic purpose hardening techniques may have limited effectiveness in preventing complex error modes, particularly due to the high number of transistors integrated in such devices. To deal more efficiently with errors in microprocessors and enhance radiation hardening, it is needed to determine how radiation can affect their operation and develop specific, microprocessor-oriented, fault-tolerance techniques.

2.6.1. Fault-tolerance techniques for microprocessors

Particular features common to microprocessors allow that specific hardening techniques can be developed to deal with faults. Microprocessor-specific hardening techniques are typically classified regarding the scope of application in software, hardware or hybrid techniques.

2.6.1.a Software fault-tolerance techniques

Software techniques derive from the concepts of software redundancy, information redundancy and time redundancy, conforming a design paradigm known as Software Implemented Hardware Fault Tolerance (SIHFT). Software techniques introduce changes in the executed code of microprocessor-based systems to improve their overall dependability

while maintaining original functionality. Most existing software techniques have been conceived to be systematically implemented in order to be used in different applications. Since the manual introduction of software techniques can be prone to mistakes, the most convenient approach is to develop a method to automatically implement the techniques in the code, commonly at software compilation time. Typically, software modifications introduced by these techniques are not specifically needed to perform any system service, but only to detect errors produced in the execution and, eventually, perform corrective actions. As a result, the code size and complexity is commonly higher compared to what is strictly needed to comply with functionality requirements, but these techniques contribute to lower the error rate and comply with dependability requirements. As a result, the software becomes a critical part of the system, as it must be effective in coping with errors. Software techniques are effective, versatile, and can be implemented at very low cost, since they do not need to modify the system design nor the hardware. However, they commonly introduce penalties in terms of higher memory use, and lower efficiency and performance due to the usage of additional, non strictly needed code. In addition, software-based techniques are limited as they can only detect error in the microprocessor resources that can be accessed by software. To overcome such limitations, different software techniques can be combined, even with other kinds of techniques, to tackle different types of errors. Software techniques are generally classified according to whether they are focused in dealing with control-flow errors or data errors [57].

2.6.1.a.1 Software control-flow checking techniques

Microprocessors are designed to manage information by executing a sequence of instructions one after another. However, to allow performing more complex tasks, some instructions may trigger branches in the execution when evaluating a condition or running in a loop. At the branches, the sequential execution flow can be altered and the processor may not execute the instruction after the branch instruction, but another one pointed by it. Software control-flow checking (CFC) techniques are devoted to check that the instructions are executed by a processor in the correct order. A control-flow error (CFE) is produced when the processor executes an incorrect sequence of instructions due to a fault. Control-flow errors represent a threat for the dependability of critical microprocessor-based systems since they can result in erroneous outcomes or even for the processor to lose the control of the system.

Since branches are the points in which the sequential execution flow is modified, most CFC techniques are devoted to check that branches in a software execution are correctly performed and that no unexpected branches occur. Control-flow checking techniques generally consider the application software as a sequence of branch-free intervals (BFIs), also called basic blocks (BBs). A basic block is an instruction or a group of consecutive instructions that are always executed sequentially in the absence of errors. In other words, none of the instructions within a basic block is allowed to execute a branch except,

eventually, the last instruction; and none of the instructions within a basic block is allowed to be the destination of a branch except, eventually, the first instruction [57].

To be applied, CFC techniques rely on the creation of a graphical representation of the control flow, called control flow graph (CFG), in which basic blocks are represented as nodes that are connected by arrows representing allowed branches according to the execution flow of the program in the absence of faults. Legal branches are those represented in the CFG, and illegal branches are those which are not. However, not every legal branch may be correct, such as the case in which a conditional branch is taken due to a fault in the evaluation of the condition. In addition, it could happen that faulty branches occur not only between different BBs but inside the same BB [57].

Most relevant software techniques for control-flow checking are CCA [58], ECCA [59], CFCSS [60], YACCA [61], and CEDA [62]. All of them introduce new dedicated data into the original program to track the control flow, usually named signature. Signature value is updated during execution by instructions added for that purpose. Checking instructions, also called assertions, are introduced to determine whether the values are correct in specific points in execution. This common practice is known as signature monitoring and can be implemented in different ways.

In the case of the high-level control-flow checking approach using assertions (CCA) [58], two data values are used to track and check control-flow correctness: the block identifier is updated at the beginning of each BFI and checked at the end to detect illegal branches from or to the middle of a BFI; and the control flow identifier is updated at the end of a BFI and checked at the beginning of the following BFI to detect illegal branches between BFIs. However, it can happen that illegal branches between BFIs are not detected in the case that those BFIs are children of parent BFIs with another common child due to aliasing in the control flow identifier [59]. Additionally, erroneous conditional branches and illegal branches within the same BFI are not covered by this technique. The authors use a pre-processor to analyze the syntactic structure of a program written in C language and to insert the data and instructions to check the control flow. As an advantage of this technique is its simplicity and portability between different microprocessors since it is directly applied on the source code. Nevertheless, CCA technique involves high memory and time overheads since the amount of identifier values increases for large and complex programs and the instructions to update and check the identifiers can become long when inserted in small blocks, thus increasing the probability of the program to get control-flow errors.

Enhanced Control-Flow Checking Using Assertions (ECCA) [59] is an improved technique based on CCA. Similarly to CCA, it features high portability by using a pre-processor to introduce new data and instructions in the high-level C program source code, although it can be also applied at a lower level by modifying the compiler. ECCA can be applied to different levels of granularity by grouping BFIs into blocks containing more than one BFI given that the block has a single entry point and a single exit point.

The size of the block may impact on the error detection latency and also the performance and memory overhead, so for maximal coverage a block may contain only one BFI, that will be associated with maximum overhead. Unlike CCA, ECCA introduces only one variable to be updated according to prime number rules: each block is assigned with a prime number identifier and its value is updated throughout execution. The set of identifiers of allowed destination blocks are multiplied at pre-process time so a division of that number by the block identifier will identify legal and illegal branches by divisibility rules. ECCA technique overcomes the limitation of CCA by avoiding aliasing and is recognized by its high CFE coverage. However, ECCA is still not capable of detecting erroneous conditional branches nor illegal branches within blocks. Moreover, despite ECCA concept and implementation is not complex, identifier management instructions and memory requirements for storing identifier values can produce high performance and memory overheads.

The technique called Control Flow Checking by Software Signatures (CFCSS) [60] also makes use of a special value, called signature, which is stored in a dedicated processor register to track the control-flow during execution. For that purpose, arbitrary numbers are assigned to each BFI at compile time and new instructions are introduced at assembly level at the beginning of each BFI to update and check the value of the signature in each BFI. In the absence of faults, the value of the computed signature must match the value assigned to the BFI which is being executed. The problem of aliasing is tackled in the technique by adding an additional value to compute the signature in those BFIs that have multiple parents and at least one of the parents has multiple children. Although the problem of aliasing is reduced it is still possible that aliasing occurs for BFIs sharing parents with other BFIs with multiple parents. Errors involving faulty conditional branches, or illegal branches within a BFI are not handled by this technique. An important advantage of this technique is that, once a CFE is produced, the signature value becomes invalid for the rest of the execution, so the amount of signature checks can be reduced to minimize performance overhead at the expense of a higher error detection latency. The performance overhead introduced is lower than other techniques since the operations with signatures are fast, but for programs with lots of short BFIs, the program size can be affected considerably. Moreover, the memory needs for storing all node signatures can also become an issue.

The Yet Another Control-Flow Checking using Assertions (YACCA) technique [61] analyzes the software to build a program graph and identify all BFIs on it. A global variable is used to store a signature value which is updated during execution. New instructions are introduced at the C source code to check the signature value and determine if the current BFI has been reached from a legal BFI and also to update the signature according to the current BFI. The novelty of the YACCA technique is that the new instructions are introduced both at the beginning and at the end of each BFI increasing fault coverage. In addition, faulty conditional branches can be detected through the introduction at the beginning of each target BFI a replica of the same condition that was evaluated before reaching it. If the two evaluations of the same condition produce different results, then a

control flow error is detected. The YACCA technique optimizes signature evaluation by avoiding division operations and performing multiple comparisons instead. To update the signature value during execution, the technique proposes the use of two constant masks, whose value depend on the signatures of possible parent BFIs and the signature of the current BFI. The YACCA method was found to achieve higher error coverage than ECCA and CFCSS techniques, despite it introduces higher memory and performance overheads than CFCSS.

Control Flow Error Detection Through Assertions (CEDA) [62] follows the path of previous techniques and uses a run-time calculated signature which is checked and updated by new instructions introduced automatically by a modified C language compiler. The main novelty of this method is to introduce a set of rules to assign signature values, allowing to reduce the check and update instructions to just one instruction with no aliasing. Two constant signatures and two constant parameters are assigned to each BFI. The run-time calculated signature is updated at the beginning of the BFI and at the end of the BFI by using the node parameters according to the rules of the technique. Signature values can be checked afterwards to detect control-flow errors, however, it is possible to reduce the number of checkpoints to reduce performance overhead at the expense of higher detection latency. CEDA technique has high relevancy due to a very high efficiency delivering high error detection capabilities with very low performance overhead compared to other techniques. However, CEDA is not capable of detecting erroneous conditional branches nor illegal branches within nodes.

Inverted Branches, as referred in [63], is a software technique for control flow checking derived from [64]. This technique is different from the previous ones in the sense that it does not introduce additional data in the program to check branch correctness, since it is only devoted to check for control flow errors related to faulty evaluation of conditional branches. To achieve that, this technique consists in replicating the conditional branch instruction by adding additional conditional branch instructions either in the true and the false control-flow path. The term inverted comes from the idea of introducing the new conditional branch instructions with the evaluation of the complementary condition so, if both branches are taken, the risk of faulty evaluation of the condition is minimized. It could be similar to YACCA in the sense of checking twice the conditional branches, but in the case of this technique, the checking is performed before taking the branch, while in YACCA the second evaluation is performed after taking the branch.

Several control flow checking SW techniques have been introduced, remarking benefits and limitations of each one. Signature monitoring techniques are effective to detect a wide range of control flow errors, but they introduce memory and performance overheads. Efficiency can be achieved by performing an optimal selection of signature values and checking instructions, by reducing the number of checkpoints at the expense of higher detection latency, or by reducing the complexity of the technique at the expense of lower error coverage.

2.6.1.a.2 Software data checking techniques

Microprocessors perform operations on data as part of the information management process in any SW application. In the presence of faults, data contents may become wrong and produce erroneous results, even in the case that the execution flow has suffered no changes. Data checking techniques are generally based on information and time redundancy combined with consistency checks that are additional instructions introduced to evaluate redundant results. These techniques are able to detect and eventually recover from data errors. However, they usually introduce significant performance and memory overheads.

In [64], a set of rules is defined to transform the high-level source code, achieving a highly general and portable solution with high error detection potential regardless of the target hardware. The rules are conveniently defined to allow a straightforward implementation that can even be automated [65], thus reducing the user effort and avoiding mistakes. The transformation rules require to duplicate every variable in the source code, so every write operation to any variable must be duplicated and performed in both copies. Additionally, consistency checks must be added after every read operation to any variable to verify that both copies hold the same value and call an error routine in the case of any inconsistency. It is remarkable that read operations correspond also to the evaluation of the value of the variable in conditional expressions. Variables that are no longer read further in the program are not checked anymore, since a faulty value on them cannot produce errors. This technique achieves a very low error detection latency and minimizes error propagation through the program. However, it introduces high memory overhead by a factor of 3 and similar performance reduction [65].

However, not all data managed by a microprocessor is equally relevant in the dependability of the resulting application. In [66], all variables in the program are duplicated, as in [65], but they are then classified into two categories as intermediate variables or final variables according to whether they are used to compute other variables or not. An automated tool is developed to analyze dependencies between variables and to introduce code transformations according to technique rules. Final variables are the only ones which are checked for consistency as they are considered the outcomes of the program, thus reducing the performance overhead with respect to [65] at the expense of increasing error detection latency. The work developed in [67] address the replication of only specific data inside a software program by the use of an automatic tool that analyzes the high-level source code to reorder instructions and selectively duplicate variables to increase dependability while minimizing overheads and preserving the original functionality. The automated tool automatically detects the variables and code regions that may more critically affect the program dependability, and allow the user to adjust a trade-off between amount of duplication and performance and memory overhead, maximizing duplication benefits. In [68], microprocessor core registers are selectively duplicated by using an automatic tool at assembly code level. The importance of register selection is assessed by evaluating the

error coverage and performance degradation when duplicating each register individually. With the obtained information, an optimal selection of registers to be duplicated can be performed according to dependability requirements. Authors demonstrate that high error detection rates can be achieved by protecting a reduced portion of registers, given that they are the most critical on the application.

The Error Detection by Duplicated Instructions (EDDI) technique [69] is based on similar concepts than previous techniques, but exploits instruction-level parallelism existing in modern processor architectures to minimize performance overheads. In this technique, variables and registers are duplicated at assembly code and new instructions are carefully scheduled to be executed using idle processor resources as much as possible. As a result, EDDI achieves low error detection latency with high fault coverage and medium performance overhead.

The SWIFT (SoftWare Implemented Fault Tolerance) technique [70], is based on EDDI technique for data error detection. As in EDDI, variables and registers are duplicated and replicated instructions are scheduled to reuse idle resources and reduce overheads by instruction-level parallelism. Checking instructions are also introduced to verify correctness of results. However, SWIFT optimizes EDDI technique by eliminating the need for data memory duplication, since it assumes that the memories are protected by EDAC, as they commonly are in modern processor architectures, so data is only duplicated when loaded from memory. It is remarkable that SWIFT still increases significantly the program memory size due to the additional replicated instructions and checks. In addition, SWIFT also introduces CFCSS [60] technique for control-flow error detection. Additional optimizations on the implementation of SWIFT allow very high error coverage with reduced overheads when compared to other techniques. The SWIFT-R technique [71] extends SWIFT to provide not only error detection but also error recovery capabilities. By SWIFT-R technique, variables and registers are triplicated, instead of just duplicating them. In the presence of single faults, correct results can be obtained by simply performing majority voting of the three obtained results.

While previous techniques focus only on replicating instructions, it is also possible to introduce replication at higher abstraction levels, such as procedure level. The Selective Procedure Call Duplication (SPCD) technique [72] achieves error detection by duplicating the calls to procedures with the same parameters and comparing between results. Among various advantages, this approach has very low impact on code size since the procedure code is untouched and it only adds the duplicated call and one consistency check. With such small code modifications, this technique is able to detect a wide range of faults that may impact on the result of the procedure without affecting error detection capabilities, but at the expense of higher detection latency. The SPCD technique also supports the user to tune the detection latency against overhead to automatically apply the duplication at procedure or instruction level, or a combination of both. The Trikaya technique [49] uses a similar approach to SPCD and additionally introduces error correction by triplicating data and procedure calls, instead of just duplicating them. Every procedure in the program is

called three times, each one with a different set of triplicated parameters and variables, and majority voting is performed between the obtained results to mask any single fault that may have occurred. The Trikaya technique has been evaluated under radiation experiments and have been found to decrease error rates by up to one order of magnitude.

At higher abstraction levels, the complete program can be replicated in processors supporting Simultaneous Multithreading (SMT) [73], [74], by executing two redundant copies of the program simultaneously in different threads and comparing between results, which is known as Redundant Multithreading (RMT) [75]. While SMT executes both redundant threads in the same processor, it is also possible to execute each thread in a different processor exploiting modern Chip Level Multi Processing (CMP) architectures [76], however this approach present challenges such as inter-processor communication to compare between results at checkpoints. Additionally, RMT and CMP can be combined as in [77] to achieve a triplicated redundant execution including recovery capabilities with only two processors.

The program duplication techniques have an interesting variation in the techniques based on data diversity. Data diversity techniques are based on running two program copies with the same functionality but using different data. In particular, the data used by the redundant copy is the same as the original one, but multiplied by a constant factor. At the end of the execution of both redundant program copies, the result of the redundant copy is checked to be the same of the original multiplied by the factor. This approach is especially interesting to detect not only transient faults but also permanent faults that can be masked when computing a particular data value but revealed when computing a different one, although it may produce overflow in calculations if the factor value is too high. The work on [78] demonstrates that different factor values maximize error detection in different functional units of the processor. It is remarkable that this approach is only suitable for programs performing arithmetic operations but not logic operations or exponential or logarithmic functions.

While all the previously explained techniques in this section rely on duplication of code and data to detect data errors, this is not the only valid approach. Techniques based on Executable Assertions, such as the one presented in [79], can also detect data errors in software programs without introducing duplicated instructions or data, but only by performing checks on computed values according to a set of rules. To achieve error detection, variables are classified among several categories such as discrete, continuous, random, monotonic or dynamic and, for each variable, instructions are introduced in the program to check whether its value is valid or not. Such checks can be boundary values or rate limitations for the value of the different variables in the program. Authors in [79] demonstrate that a very high error coverage can be achieved with this technique. However, the use of assertions to perform data error detection is highly application-dependent, as each application may have different constraints for each variable, and requires deep knowledge on the system to be correctly implemented.

2.6.1.b Hardware fault-tolerance techniques

Hardware techniques rely on hardware redundancy concepts to increase the dependability of a microprocessor by modifying the original circuit and/or adding new hardware to it. Hardware redundancy can be introduced at transistor level by applying traditional TMR to mask errors as in any integrated circuit. However, systematic hardware replication introduce notable impact in circuit area, cost and power consumption, that possibly could not be afforded by cost- or power-constrained applications. In such cases, trade-offs should be posed to determine the optimal amount of replication for maximum error coverage with minimum penalties. In that scenario, the techniques based on Partial TMR [80] could be particularly interesting by selectively replicating the most critical processor resources or functional blocks [6]. Approximate TMR [81] is a recently introduced extension of Partial TMR in which the hardware is replicated with circuits that perform a similar function to the original one, but requiring a lower amount of area, looking for an optimal balance between error coverage and overheads.

Despite fine grain TMR-based techniques can provide convenient fault detection and masking in microprocessor-based systems, all of them require the user to have detailed knowledge about the architecture, which is not commonly available, and access to the hardware design to modify it, which is also expensive. To overcome this limitations, it is also possible to implement coarse grain redundancy at processor level by using more than one redundant processors executing the same application simultaneously in a synchronized manner, which is called lockstep. Outputs from two processors can be then compared in a cycle-by-cycle basis to detect errors with a Dual Core LockStep (DCLS) configuration and restore the system back to a previously saved error-free state [82]. However, system restoration can introduce performance overheads in systems with strict time constraints. In this case, Triple Core LockStep (TCLS) can be applied, in systems where three processors are available, to mask single errors by adding majority voting and synchronization hardware [83].

Redundancy schemes cited above need either access to the processor architecture to introduce modifications or the hardware to already embed redundant resources such as multiple processors to configure error detection or correction schemes. In the case that the user cannot introduce redundancy in the processor due to technical constraints or budget limitations, it is also possible to increase the dependability of a processing system by adding external hardware modules, which are often referred to as watchdog processors. Watchdog processors must be connected to pre-existent processor architectures through available connection points which are critical to determine the amount and type of accessed information, and thus the error checking capabilities of the watchdog processor. Watchdog processors may work in parallel with the main processor to perform four monitoring approaches: (1) check memory accesses, (2) check data consistency, (3) check program control-flow, or (4) detect symptoms of faults [84]. The operation of watchdog processors can be intrusive if they are capable of interfering the processor operation, or non-intrusive

if their functionality does not affect the processor execution. Typical implementations of watchdog processors are listed below:

- Program timer: The simplest form of watchdog processor is a watchdog timer [33] capable to detect whether the executed application has an error which impedes its finalization. Watchdog timers are typically included in processing systems and are designed to count down in every clock cycle of the processor. If the processor is executing properly, then the application may reset the count of the timer before it reaches zero. When the processor fails in a manner that loses control of the application, then the application shall not reset the count and the watchdog timer will reach zero, becoming the symptom of a fault. In that instant, the watchdog may automatically perform a reset of the entire processing system to put it back in a working state.
- Memory checker: A very typical connection point for watchdog processors is the memory bus, either the program memory, the data memory, or both, whenever they are available. In [63] a watchdog processor is implemented to check that the processor is accessing allowed memory regions, the consistency of given data variables in the executed program is correct or even that the branch instructions are correctly executed. In [85] and [86], the watchdog processor creates a shadow copy of the data in the program in a local memory to update or check it whenever is written or read, respectively, and additionally it monitors the control control flow of the program by calculating static signatures of executed instructions inside each BFI and also by validating the order or execution of BFIs according to the Program Graph.

Generally, hardware techniques can provide high error coverage, which is linked to their access to the processor architecture, or in the case that it is not possible to the amount and quality of information available at their connection point. Additionally, hardware techniques are usually associated to low performance overheads since the redundant hardware performs concurrently with processor execution.

2.6.1.c Hybrid fault-tolerance techniques

Hybrid techniques combine the approach of both software- and hardware-based techniques to optimize the achieved hardening level by leveraging the best qualities of them. Typically, the most attractive features of software techniques are their low implementation cost, flexibility and their suitability to deal with data errors. However, pure software techniques have been found to have high memory and performance overheads and also present low detection rates on control-flow errors [87]. Conversely, hardware techniques are appreciated by their low impact on system performance and their concurrent performance, resulting convenient for detecting control-flow errors [63]. Nevertheless, techniques based on external watchdog processors are limited to the available information

at the connection point, so errors occurring in a region that cannot be checked could be left undetected [88].

The approach presented in [84] is oriented to detect data errors by the combination of both software modifications on the high-level code to duplicate data and instructions and the introduction of a dedicated hardware module connected to the system bus to check data consistency. In addition, new instructions are added to the application to allow the external module to identify the values for checking using a checksum value. In the case that the checksum value is found to be incorrect by the external module, it signals this condition to the processor to activate a recovery procedure, in which the faulty variable is located and corrected, allowing the application to continue without errors. This approach was tested under fault injection and was able to detect and correct a high amount of faults while maintaining low memory and performance overheads.

In [88], a hybrid approach called HETA (Hybrid Error-Detection Technique Using Assertions) is used. HETA combines the use of software modifications and an external hardware module is used to detect control-flow errors in a processor. The overall technique derives from CEDA technique [62] and is intended to reduce its performance overhead while increasing its error detection capabilities by introducing checks in the software side for the errors that cannot be detected by the original technique, such as incorrect conditional branches through the application of the Inverted Branches technique. The software control flow checking technique is complemented by duplicating variables and instructions and introducing consistency checks to detect data errors. The hardware module provides detection of additional error types such as incorrect branches to unused memory addresses and infinite control-flow loop conditions. Additionally, the external module performs consistency checks in the software control-flow signature value. HETA was evaluated under fault injection and exhibited full control-flow error coverage as well as reduced size and performance overheads.

2.6.2. Fault-tolerance techniques for COTS microprocessors

The use of COTS-based processing systems in space applications has been an evolving topic for decades due to the constantly driving needs for higher computing performance at lower power consumption levels, resulting in more capable, more powerful, more autonomous and more efficient spacecraft [89]. In fact, radiation hardened (rad-hard) processors have been found to lag several generations behind the processing capabilities of emerging COTS processors [6], [90], [91]. However, to safely introduce COTS devices in space missions, their inherent susceptibility to radiation-induced faults needs to be mitigated [89], [92].

In [93], three strategies are presented to overcome the limitations of currently available processing systems for space:

- Developing and building a completely new processor for space use using RHBD and RHBP techniques.
- Building a RHBP version of an existing COTS processor by migrating the processor design from the commercial technology to a radiation hardened technology.
- Hardening existing COTS processors at board and/or software levels.

However, the first and the second approaches would be affected by development and qualification processes that may impact their time to market. Moreover, the outcome of such strategies may get outdated as new generations of commercial devices arise [6]. Conversely, the latter approach can be oriented in a more general manner to develop techniques that, once applied to unmodified COTS processors may improve their performance under radiation. Such techniques would then be applicable to a wider range and even successive generations of COTS devices, remaining relevant over time.

Nevertheless, hardening unmodified COTS devices introduces limitations in the fault mitigation possibilities. As the device cannot be modified, RHBP and RHBD at device level cannot be applied. Considering that COTS devices are not commonly produced with specific methods to reduce their susceptibility to radiation, the application of fault avoidance approaches is limited to device screening to handle manufacturing variability and checking device susceptibility through radiation testing [48]. However, COTS devices must be considered to be susceptible to radiation effects and user-introduced fault tolerance must be considered to cope with them. Given that the device cannot be modified, fault-tolerance approaches for COTS processors are usually based on software techniques or external hardware modules (watchdog processors).

COTS processors have already been successfully introduced in space missions. In [94], a COTS-based computer was introduced in a spacecraft in combination with another radiation hardened processor, both performing similar tasks. The performance of both processing systems was evaluated and compared in orbit. It was observed that the COTS-based computer experienced a higher amount of faults than the rad-hard one. However all observed faults could be mitigated with software error detection techniques (EDDI and CFCSS) and a basic recovery mechanism consisting on a processor reset.

Fault detection and recovery is a common approach to introduce fault tolerance in devices not supporting fault masking nor fault avoidance techniques, such as COTS-based processing systems whose hardware cannot be modified.

2.6.2.a Microprocessor fault detection and recovery approach

Fault detection and recovery is a process by which the system needs to recognize that a fault has occurred in order to trigger a subsequent action to mitigate the fault before it becomes an error. In a successful fault detection and recovery implementation, faults should be detected, located, isolated and removed from the system before producing errors,

obtaining a fault tolerant system [9]. The effectiveness of the recovery step would then strongly depend on the amount of available information about the error to take the most convenient corrective action.

Fault detection alone cannot reduce the amount of errors in a system, but plays a critical role in reducing the risks associated to the error as it can be signaled to other systems and mitigated at higher level. Undetected errors, conversely, cannot provide any risk information to the system, leaving it completely vulnerable [41]. It is widely accepted that fault detection is the first step to fault tolerance [3], [70], [74], [84], [95]–[97]. Error detection should pursue to have high coverage, to leave almost no undetected errors, and low latency, to detect errors shortly after they are produced, thus leaving few time to produce undesirable effects and enabling straightforward mitigation strategies [95].

Fault detection techniques can be classified into four main categories [96]:

- Periodic built-in self-test approaches, leveraging internal self-test mechanisms already available in processors, which are traditionally devoted for manufacturing testing. These tests are envisioned to detect permanent faults since they are not executed concurrently with the user application.
- Dynamic verification approaches consist on the application of consistency checks, typically through dedicated hardware to verify execution correctness.
- Anomaly detection approaches intend to detect anomalies in execution, or symptoms of faults, that are observable with simple monitors, such as program crashes.
- Redundant execution approaches, which exploit the replication capabilities of processors in multi-core or multi-threaded architectures to execute two or more copies of the same program and compare results.

A detailed and extensive review of fault detection techniques can be found in Section 2.6.1 in this document.

The rich variety of fault detection approaches and fault models makes it necessary to develop appropriate fault recovery strategies in each case. Fault recovery techniques can be classified into two broad categories [96]:

- Backward Error Recovery (BER), in which the state of the system is periodically saved in an error-free execution, creating checkpoints representing correct system states. In the case that an error is found, the system state is restored from a correct checkpoint in a process known as rollback.
- Forward Error Recovery (FER), by that the error detection and correction are performed through redundancy schemes, such as TMR, without requiring any checkpointing nor rollback actions.

The most simple error recovery technique is just to restart the faulty program from the beginning, as implemented in [94], typically by a system reset, to remove all faults and put the system back to an initial error-free state. This technique would be able to remove every detected error with low development effort. However, the main drawback of this technique is the decrease of performance and system availability due to the time required for the system to recover the intended functionality after a restart is made.

When the system is required to have high availability, the optimal option is to use a triplicated schema, such as the presented in [83]. In it, a commercial ARM Cortex-R5 processor is implemented in triple core lockstep (TCLS) architecture, achieving global fault tolerance without rollback even when each individual core is susceptible to faults. Fault tolerance is achieved by the implementation of a majority voter, hardware error detection and specific-purpose synchronization logic to restore state of the faulty core with the state of the two fault-free cores. The main advantage of this approach is that the recovery action is transparent to the user, however it requires to use a specific device supporting the mentioned architecture.

In the case that the available processing system has less than three cores and/or it lacks of error detection, voting and synchronization built-in logic, then the recovery schemes have to be customized and developed according to the target system and requirements. An additional precaution is to avoid the fault-tolerant system to output any data or action before checking the correctness in the data and /or the execution flow [74], [76], [97]–[99] to prevent failures.

Combined checkpoint and rollback strategy is a commonly used approach to deal with both data and control-flow errors. Data errors can be recovered by duplicating data, in any single- or multi-threaded systems, and to create checkpoints after successful consistency checks. The control-flow of a program can be typically validated in single-threaded systems by the use of signature checking [99], [100] and by comparing the execution-flow of multiple program instances in multi-threaded systems [74], [76]. In both cases, checkpoints may be generated after each successful check. In the case of a failed consistency check, the system may be rolled back to the previous checkpoint before resuming operation. For this approach to be successful, it is needed that not only processor state is saved at the checkpoint but also all program data relevant to execution. Data protection can be applied extensively to the whole program [84], [96], [99] or selectively to only certain critical points [97], [98]. The main advantage of Checkpoint and rollback strategy is that it can be adapted to a broad range of systems with low to moderate performance overheads, provided that the frequency for recovery actions is low, and still achieve a good error coverage.

An alternate solution is to artificially recreate the Triple Modular Redundancy (TMR) behavior in a system that does not include it by default. The SWIFT-R [71] technique achieves high error coverage in single-threaded systems with low user effort, at the expense of significant performance reduction. Alternatively, the approach presented

in [77] leverages a multi-threaded system to perform duplicated program execution and trigger a third execution of the last program segment upon an error. With the three program outputs, voting and synchronization routines are performed to mask the fault and continue execution.

2.6.2.b Microprocessor error diagnosis

Error diagnosis, as referred in this section, is the process of determining the circumstances, mainly time and location, in which the causes of a subsequent error occurred. A successful diagnosis process should match most errors and causes correctly, however, the diagnosis process cannot start until errors are initially detected. As a consequence, a very high error detection coverage is needed to achieve high standards of diagnosis. In fact, error diagnosis can be seen as an intermediate step between error detection and recovery [96], as it can provide valuable information to the error recovery process about the faulty event to apply an efficient recovery strategy. However, error diagnosis in microprocessors typically becomes a high uncertainty process due to the inherent complexity and low observability of processing architectures.

A common approach for error diagnosis is to detect errors based on their consequences and then try to perform a deduction of the fault that caused it. The main drawback of this approach is that a fault in different locations may produce similar consequences, which is known as fault aliasing [101], [102], limiting the accuracy of diagnosis approaches based on cause-effect analysis. To cope with this problem, researchers have proposed to study the behavior of systems under intentionally provoked and controlled faults, to retrieve information that may be used afterwards to diagnose uncontrolled errors [103], [104]. For such purpose, broad beam radiation testing is not a suitable option, since it provokes faults at random locations and instants. Instead of that, techniques such as software-based fault injection [56], hardware-based fault injection [105], laser testing [42] or micro-beam techniques [106], [107] may be used as they provide good control on the location and the time that the fault is provoked.

Error diagnosis for microprocessors has been a matter or research since long time [108], but it was mainly devoted to identify permanent errors associated with manufacturing or device wear-out. However, soft errors are becoming more frequent due to the progressive miniaturization of electronic devices, and there is a growing interest in developing diagnosis techniques addressing them.

A common approach for error diagnosis is the creation of error dictionaries [101], [104], [109] based on device responses under extensive fault injection sets. The error dictionary is built to relate the effects of the error to its possible causes, so when a faulty device behavior is found, it could be associated to a set of possible causes using the information at the dictionary. A common problem with error dictionaries is that they can become very big for an exhaustive set of injected faults in a complex device. In [110] manufacturing defects on a chip are diagnosed through the creation of an error dictionary that relates fault

locations with circuit responses based on the results of extensive fault injection, and an associated computational tool is developed to optimize the diagnosis process. The work in [104] performs an extensive hardware-based fault injection in two different circuits and introduces a compression technique to the circuit responses to reduce the dictionary size. This work is continued in [101] by developing metrics to identify the set of input vectors that minimizes the aliasing introduced by compression and applying them to an error dictionary obtained from hardware-based fault injection on a microprocessor unit.

Other important approach for error diagnosis is the use of device self-testing strategies, which fall in the Design for Testability (DfT) paradigm. DfT techniques are commonly implemented to check the health status of a particular unit in different scenarios, from manufacturing acceptance tests to capability checks while on operation. In this context, diagnosis is also relevant to improve maintainability of systems, as it can reduce the time required to determine the cause of the error. In [111], a processor is designed to be capable of self-diagnosing and self-repairing from hard-faults by a reconfiguration strategy in which internal modules accumulating a high number of faults in a short period of time are automatically turned off. The processor used in [112] include an embedded mechanism called Machine Check Architecture (MCA) which enables detection, location, isolation and correction of soft errors and, ultimately, error diagnosis. In [102], scan-based Built-In Self-Test hardware available in the device is used to diagnose faults on chip through the generation of automatic test patterns and checking circuit responses with a fault dictionary.

However, the use of dedicated hardware to diagnose errors in a circuit presents, two main drawbacks: first, that the diagnosis hardware can be vulnerable to faults, which may be left undiagnosed; and second, that it produces area overhead so it is required to be very compact. To overcome these limitations, software-based self-test approaches can be proposed. Such tests are commonly used in critical systems to check the health status of the overall system before committing a critical task. In [108], diagnostic software programs are proposed to be run on a microprocessor to check the correct operation of internal units. In [113], a set of systematic techniques is presented to identify faults present in different processor sub-modules. The approach presented in [114], is able to diagnose and repair permanent faults on a microprocessor on the field through a mixed approach combining an initial hardware-based self-test to check basic processor functionalities and two software-based self-test procedures to check the processor slots and the register file.

Artificial intelligence methods have also been proposed to address the problem of relating the detected wrong behaviors with the parts of the device affected by faults. A review of these methods, such as expert systems, Petri nets, neural networks and fuzzy logic is presented in [115].

2.6.2.c Trace interface-based fault-tolerance

Among the diverse fault-tolerance techniques applicable to processors, two main general drawbacks can be identified:

- Software techniques generally introduce undesirable performance overheads.
- Some hardware techniques require to modify the device, which cannot be applied to COTS.

In such context, providing effective fault-tolerance techniques against soft errors with low impact in performance and minimal intrusiveness in the device is a challenge, which has become an important research topic. The main approach is to develop hardware modules, also known as watchdog processors, that can be connected to existing processing architectures through suitable interfaces to monitor the processor operation in a non-intrusive manner. Hardware monitoring approaches commonly introduce low or even no performance penalty in the system, as they run at high speed in dedicated hardware concurrently with the processor. The main objective is to observe the internal processor behavior to identify the errors before they manifest externally and become failures. However, the connection point for the hardware module to the processor is an outstanding issue, as it may become invasive and interfere with the normal behavior of the circuit. The most convenient observation point could be the memory interface, as it is the same that the processor uses to fetch instructions or store data, but it is not commonly available in modern and complex COTS processors with integrated On-Chip Memory (OCM) and cache memories. Moreover, for microprocessors with complex pipelines, the information available at the memory interface may not accurately reflect the actual processor behavior, as fetched instructions may not be executed due to speculation [1].

It is worth noting that processor observation and monitoring capabilities are also crucial features during software developing and debugging processes at early production phases. Such functionalities are supported by On-Chip Debugging (OCD) interfaces, which have become mainstream in modern devices. While debugging interfaces provide useful information about processor operation, they are generally left unused once the application development is complete [2], so they can be reused for hardware monitoring purposes in an inexpensive way. Among different resources associated to OCD infrastructures, the trace interface is particularly suitable for processor observation. It is commonly conceived as a read-only interface used for software debugging and application profiling and can provide information about processor execution without affecting it or provoking any disturbance. Unlike the memory interface, the trace interface accurately reflects the actual processor behavior, by only exporting information about the executed instructions. A wide overview on the use of the trace interface for microprocessor observation is presented in [1].

Hardware monitoring through the trace interface can solve the observability challenges related to complex processing architectures [33], especially when the access to the internal resources is limited. Trace interface implementations in different processors tend to be comparable, as they are built for the same purpose, and provide similar information that can be used for fault detection and diagnosis. Typically, it may contain information about the executed instructions, program counter, status flags, or memory accesses [1].

Fault-tolerance approaches based on the trace interface may increase the dependability of microprocessor-based systems while being general for different types of microprocessors

The development of watchdog processors for hardware monitoring through the trace interface has been mainly oriented to detect control-flow errors on soft-core processors [1]–[5], [7]. The focus on control-flow errors can be justified in the sense of need, by the challenges presented by software techniques to deal with this type of errors and, at the same time opportunity by the fact that the most common information available at the trace interface is related to program execution, particularly, to the instruction address of the processor. Data checking techniques can be then efficiently added by software if needed, conforming a hybrid approach. The preference for soft-core processors lies in the richer possibilities for connecting the hardware modules and developing a custom solution, while in the case of using commercial hard-core processors, the system design and the access to the trace interface information may be more complex due to the existence of proprietary interfaces or encoding protocols.

Hardware monitoring through the trace interface was proposed in [2]. In this work, two LEON3 processors executed the same software asynchronously. A watchdog processor module was developed and connected to the trace interface of both processors to receive execution information, composed by the values of the program counter, the operation code, the instruction data and processor flags. The information obtained from each processor is compressed by the module to compute a signature for each processor. Every time that a processor finishes a task, the task signature is saved so when the other processor finishes the same task, both can be compared. In a fault-free scenario, both signatures should be equal since the trace information generated by each processor should be the same. In the case that the signatures are found to be different, an error is signaled to start a recovery procedure, which can consist in the repetition of the faulty task. The main advantage of this approach is that it does not impact on performance and does not need to modify the executed software. As a disadvantage, two identical processors are required. Authors performed a fault injection campaign in the HDL netlist of the whole system and demonstrated high error detection capabilities with very low latency.

The work in [3] is based on [2] and describes more general applicability considerations of the method, by applying it to different system topologies. A hardware module, similar to the one introduced in [2] is described and used to observe the execution of processor tasks through the trace interface and compute a signature based on the obtained information. When the hardware module is introduced in a system with two processors, then it may be able to detect faults when the obtained signatures of the same task differs, as in [2]. But the approach is also extended to the case with three or more processors, in which the hardware module may gain not only error detection capabilities but also error correction with the help of a voting scheme. In the opposite case that the module is implemented in a system with only one processor, then the approach may also be feasible by replicating the execution of the tasks in different instants of time and comparing at the end. Additionally, authors also explore the impact produced by applying the same approach to two different processing

architectures: a soft-core LEON3 and a hard-core ARM7TDMI, each of them presenting a different trace interface. In the case of the LEON3, the module can be directly connected to the trace bus in a System-on-Chip configuration thanks to its soft-core implementation. Conversely, the ARM7TDMI devices is a hard-core that only allows access to a trace buffer through an external trace port. The authors tested the presented technique only for a system with two LEON3 processors through fault injection in the HDL netlist observing high fault coverage capabilities under fault injection campaigns.

In [7] a hardware module is developed for signature monitoring through the trace interface. The application code is divided into Basic Blocks (BBs) and the trace interface continuously provides the values of the Program Counter (PC) and Instruction Register. A memory block is included in the hardware module to store a table with the addresses of each BB in the application along with their signature, which can be computed at compile time. The module receives the trace information on the fly and computes the signature of the BBs under execution with low latency. When the module detects the end of a basic block, then it checks the signature value against the expected one stored on the memory and raises an error signal in the case of a mismatch. Additionally, the module can also detect the correct execution flow during a BB or in between BBs by comparing the incoming PC address with the expected one based on the previous instruction in a way similar to [4]. This technique, as any other signature-based technique, presents a disadvantage related to the memory size needed to store the signature, which can become unmanageable for complex applications. In a first approach, authors propose to select a subset of the BBs to be stored in the table by considering the size of the BB and the number of times that each BB is executed. This way, the technique may maintain most fault coverage capabilities while reducing the memory requirements by discarding those BBs which are infrequently used or that are extremely small. However, it could happen that the available memory is significantly smaller than the amount of BBs to be stored. In this case, the authors propose to let the hardware monitor the capability to dynamically determine which BBs should be stored in the memory. Thus, the memory shall start empty, and the module will introduce each BB as soon as it is executed. BBs may remain in the memory using a Least Recently Used policy, so the most frequently used BBs will remain in the memory while the least shall be overwritten, thus optimizing memory usage. In the case that a BB is executed and is already present in the memory, its signature could be checked and an error detected. Authors report that this approach may initially miss to detect some faults but still achieves a good error coverage and removes the need of pre-computing the signatures at compile time. The mentioned approaches were tested in two different processors, miniMIPS and LEON3, with fault injection campaigns.

The trace interface is also used in [4] to monitor the execution of a LEON3 processor from an external hardware module which implements a Program Counter (PC) prediction technique to detect illegal changes in the execution flow. The approach does not require any additional information to be stored or configured in the module, providing a general solution independently of the executed application. The external module is capable to

predict the next PC value based on the operation code of the currently executed Instruction Register value and the current PC value, both retrieved through the trace interface. If the operation code is a branch instruction, then the following PC value shall be incremented by the branch offset, otherwise the PC value shall be incremented by the size of the instruction. Additionally, the module is also capable of maintaining a replica of the processor stack to support the checking of the subroutine call and return statements. If at any time, the value of the PC predicted by the module and the one provided by the trace interface differ, a control-flow error is detected. The control-flow checking (CFC) technique is combined with two alternative software-based data protection techniques presented in [71] and [72], depending on the global required detection latency and introduced overheads, to conform a global hybrid technique. The complete hybrid technique achieves high fault coverage capabilities.

The technique presented in [5] is an evolution of the one introduced in [4] to overcome the limitations of the latter to detect errors in highly pipelined processor architectures. In such processors, a corruption of the PC or the Instruction Register inside the processor pipeline may not be detected by just performing PC prediction, since the faulty values shall be output by the trace interface and the processor operation based on such values may be interpreted as correct by the hardware module. To detect such errors, [5] introduces a technique called Dual Control-Flow Monitoring by that the execution flow is monitored by a hardware module both at the memory interface and at the trace interface. By comparing the instructions at both observation points the module is able to detect any discrepancy between instructions before and after they are executed, provoked by faults inside the pipeline. Authors claim that the combination of the Dual Control-Flow Monitoring technique with the PC prediction technique presented in [4] is capable of detecting all control-flow errors in a LEON3 microprocessor with no hardware modification and no performance degradation. A timeout detector is also included in the hardware module to the system to detect situations in which the processor becomes irresponsive due to a program crash. The hardware monitoring approach is complemented by software techniques consisting data duplication [64], [65] and inverted branches [63], [64], conforming a global hybrid technique. The hybrid approach was tested under fault injection and demonstrated a 100% control-flow error detection and high coverage on data errors with low overheads. The main drawback of this approach is that the required access to the memory interface may not be available in highly integrated System-on-Chip COTS devices.

The work in [8] demonstrates that a system based on [5] is capable to detect errors and improve substantially the fault-tolerance of a soft-core LEON3 microprocessor under fault injection and also neutron irradiation while running on a commercial device.

2.6.2.d ARM CoreSight Trace and Debug subsystem

This Thesis has been focused on the CoreSight trace subsystem, which is a common resource in ARM-based SoCs [116]. CoreSight combines a wide set of technologies and

architectures developed by ARM for trace and debugging purposes in high-complexity SoC designs, and is compatible with almost every processor developed by ARM [117]. In this context, debugging is associated to the observation or modification of the state of a design by directly accessing to register values and eventually halting the execution. In contrast, trace allows for collection of execution information as a continuous data stream for ulterior analysis in a non-invasive manner. The wide range of CoreSight components [118] cover several trace and debug functionalities, providing flexibility to adapt each system to a particular need leveraging a modular design. In addition, documentation is available for designers to extend CoreSight capabilities with custom new designs.

CoreSight components can be categorized into three types:

- Trace sources, are components directly connected to the processor core, devoted to organize the trace data from the processor state and generate normalized trace packets.
- Trace links, are the components devoted to gather, merge and transmit trace data generated at the trace sources.
- Trace sink components may add format to the trace data and allow it to be accessed by users on-chip or off-chip.

The CoreSight subsystem implemented in Xilinx Zynq-7000 AP SoC device family [119] can be used to illustrate the modular and flexible implementations supported by the CoreSight architecture. This device includes a dual-core ARM Cortex-A9 processor, and each core has an associated trace source, called Program Trace Macrocell (PTM) [120] to trace execution flow of instructions. An additional trace source, called Instrumentation Trace Macrocell (ITM) [118] is provided to trace application data values. Each macrocell produces trace data as a stream of normalized packets according to a protocol specification. The trace streams are merged and tagged by a trace link component, called Funnel, and transmitted to two different trace sinks. One trace sink is the Embedded Trace Buffer (ETB) implemented on-chip to store the most recent trace information. The other trace sink is the Trace Port Interface Unit (TPIU), which allows trace data to be exported off-chip through the pins.

However, not all ARM devices including CoreSight technology share the same implementation. In the case of the Xinlinx Zynq UltraScale+ MP SoC device family [121], a more powerful processing system is provided comprising a quad-core ARM Cortex-A53 along with a dual-core ARM Cortex-R5. Each of the ARM Cortex-A53 cores has an associated trace source, called Embedded Trace Macrocell version 4 (ETMv4) capable of tracing the instruction flow. In contrast, each of the ARM Cortex-R5 processors include an associated Embedded Trace Macrocell version 3 (ETMv3), which also provides execution flow information, although it uses a different trace protocol. Another difference between ETM implemenations is that the ETM modules for ARM Cortex-R5 processors can also trace load and store accesses of data values in memory while the ETM modules

for the ARM Cortex-A53 processors cannot. The CoreSight subsystem of the Zynq UltraScale+ devices is complemented with the System Trace Macrocell (STM), which is a more capable extension of the Instrumentation Trace Macrocell, also devoted to trace data values. Finally, the Funnel, the TPIU and a trace buffer are also included.

The ARM CoreSight trace interface has been used in the literature for different purposes.

The work in [3] already analyzes the possibilities of using the ARM CoreSight Embedded Trace Macrocell (ETM) trace source for transient fault detection and proposes an approach for hardware monitoring. However, the approach is not tested with the ARM trace interface, but with the soft-core LEON3 processor instead.

In [122], a hardware module is proposed to monitor the trace information generated by the Program Trace Macrocell (PTM) on a Xilinx Zynq-7000 device. In this approach, the trace information is used to increase the observability of a processing system during system tests and especially focusing to power-on self-test routines. Authors designed an IP-core to be implemented in an FPGA as a processor module capable of reading the trace from the Trace Port Interface Unit, while being configurable through memory mapped registers. The trace information generated by the PTM is processed to extract the different packets and discard a subset of them which are not deterministic. The remaining information is introduced in a multiple input shift register to generate a signature of the self-test routine, which is checked at the end to detect anomalies in the system. However, authors didn't test the proposed approach in the Zynq-7000 device, as they were not able to appropriately simulate the faults. As a result, a soft-core MIPS processor producing similar trace information was used for the tests, and the proposed approach was tested with fault simulation on the processor netlist, showing a good error coverage. It is remarkable that this approach does not need to correctly interpret the information contained in the trace data, but only that the received trace data matches the expected signature.

The work in [123] describes the development of a complex hardware module intended to improve the security of a processing system against malicious software attacks trying to retrieve critical information. While common techniques in this field are based on software and introduce very high overheads, authors propose to reuse the information provided by the trace interface to drastically reduce the impact in performance. The presented approach is developed for a Xilinx Zynq-7000 device and leverages the information about Program Counter (PC) values provided by the Program Trace Macrocell (PTM) in the CoreSight trace subsystem. Authors also introduce software instrumentation and static analysis to complement trace information to perform dynamic information flow tracking. As a result the technique can successfully detect and prevent unauthorized system accesses with reduced overheads.

In [124] and [125], authors propose to reuse the CoreSight trace interface available on a Xilinx Zynq-7000 device to protect the system against malicious attacks trying to manipulate the control-flow of a victim processing system. Authors developed a hardware

monitor capable to detect Code Reuse Attacks, in which attackers may be able to reuse existing code to perform arbitrary computations, and Code Injection Attacks, in which attackers may be able to introduce or modify the code to change the system behavior. The detection is performed by leveraging the branch information provided by the Program Trace Macrocell (PTM) source available in the system.

In [126], a complex, reconfigurable hardware monitor is developed to process trace information from the dual-core processing system of a Xilinx Zynq-7000 device in real time. The module can be reconfigured by the user by specifying the correct operation of the program executing in the processor, and produces an output that allows to debug or refine the implementation of the application. In [127] an extension of the previous module is described to be applied in a more generic manner. The module can be first configured with the Control-Flow Graph (CFG) of the executed application and is then capable of reconstructing the program execution based on the information received from the trace. Reconfigurable modules allow to recognize events in the execution, allowing to perform tasks such as code coverage measurement, worst-case execution time estimation, or finding functional or timing violations. Once the module has retrieved enough events, the generated information can be downloaded to a PC to allow further examination or post-processing of the results. The proposed approach may overcome the limited observability of modern SoCs to support application debugging and validation tasks with fast reconfiguration capabilities, achieving high efficiency.

Works such as [128]–[130], propose to store the trace generated on a buffer, such as the Embedded Trace Buffer (ETB), analyze it afterwards from the software and detect malicious attacks compromising security. This approach is limited in terms that the checking software must be protected [128]. Additionally the checking process cannot be performed online with application execution [129], increasing latency and performance overheads, although this limitation could be partially overcome by using a multi-core system and checking the trace of one core from another core [130].

The work in [131] introduces a hardware module to perform trace monitoring on Xilinx Zynq UltraScale+ device [121] and identify security-threatening attacks such as code reuse attacks. The hardware monitor leverages the instruction information provided by the Embedded Trace Macrocell version 4 (ETMv4) in the device along with the Control-Flow Graph of the executed application to compute and identify the signatures of correctly and incorrectly executed branches. The module was able to successfully detect all of the attacks performed by the author in three application cases representing different attack types.

In [132], the design of a hardware module capable of decoding ETMv4 trace protocol for both instruction and data is described with detail. Such design is later used by the same authors to develop a hardware monitor for data transfers [133] providing increased observability for online verification and debugging purposes.

The approach in [134] exports the trace information, produced by the processing system of a Xilinx Zynq-7000 device associated to an executed application, to an external host.

The host is a computer which is able to compare the branch sequence against the expected Control-flow Graph of the application to detect faults produced by environmental effects such as radiation. Authors identify that additional analysis on the trace information may allow more insight about the events, such as the type of exception that conducted to an execution timeout, or the fault detection latency. A fault injection campaign is performed in the register file of the processor and detailed error rates are given per each register while running different applications. Authors recognize that the trace analysis capabilities could be implemented on a dedicated hardware module to speed up the fault detection process. The module is then presented in [135], which is capable of decoding the trace information received from the dual-core processing system. A configuration memory in the module stores the valid sequences of branch addresses conforming the Control-Flow Graph, to evaluate the correctness of the values of the Program Counter received from the trace. The effectiveness of the hardware module under radiation is only analyzed through AVF methods, and tested under a limited fault injection campaign.

Despite the previously described works could be partially applied to the objectives of this work, almost none of them has been specifically conceived for the detection of radiation induced errors, and none of them include radiation testing results. Additionally, it must be noted that some of them have been published later than the start of this work [127], [130]–[132], [134], [135].

This Thesis is devoted to develop new error detection and diagnosis techniques for hard-core high-end COTS processors leveraging the information available at the trace interface. The ARM CoreSight trace and debug subsystem has been identified as a target technology to apply the developed techniques. It must be noted that very few documentation about the use of ARM CoreSight trace interface for custom processor observation purposes was available at the beginning of this Thesis. In fact, the most relevant documentation used during the development of the work were the catalog and data sheet documents from the manufacturers [117]–[120]. In addition, this is the first work intended to perform detection and diagnosis for both data and control-flow errors in hard-core microprocessors through the trace interface. Moreover, this work has been pioneer to provide radiation results on the developed error mitigation techniques based on the trace interface for hard-core processors.

3. MATERIALS AND METHODS

This chapter describes the material resources and the research methodology followed during this work. Due to the compendium of publications modality of this Thesis, chapters 4 to 10 already include materials and methodology considerations. However, some features are common to more than one publication and details are spread among different chapters. This chapter is devoted to integrate all relevant materials and methods to improve contextualization between them, give the reader a cohesive point of view and provide more detailed descriptions where possible.

3.1. Resources

The development of this work has been supported by the use of specific electronic devices and tools, access to specialized facilities and other auxiliary equipment.

3.1.1. Vehicle of study

3.1.1.a Zynq-7000 AP SoC

Zynq-7000 is a modern product line of devices manufactured by Xilinx. Launched in 2012 with Zynq-7000 family [119] and extended in 2016 with Zynq Ultrascale+ family [121], they conform a set of devices which Xilinx denominates All Programmable System-on-Chip (AP SoC). As a common feature, they all integrate a hard-core processing system (PS) composed by one or more ARM microprocessor cores and a FPGA programmable logic (PL) region in the same chip. The AP SoC denomination refers to that both the processing system and the programmable logic can be programmed by the user.

Zynq-7000 family has been extensively used within this work as a development platform and vehicle of study. It has been selected due to the following reasons:

- Dual-core ARM Cortex-A9 hard-core processing system (PS). This is a high-end processing architecture which presents representative elements common to other modern, high-performance processors.
- Fast and large FPGA programmable logic (PL) region, in which additional hardware modules can be implemented.
- Trace and debug CoreSight subsystem [118] including Program Trace Macrocell (PTM) and Instrumentation Trace Macrocell (ITM).

- Flexible and versatile Extended Multiplexed Input Output (EMIO) interconnection signals between PS and PL.
- Device novelty, although not immaturity at the date that this Thesis started, with high interest by the radiation effects and space communities.
- Available and affordable evaluation and development boards supporting various devices belonging to the Zynq-7000 family.
- Previous experience using these devices in the *Diseño Microelectrónico y Aplicaciones* (Microelectronic Design and Applications, DMA) research group at University Carlos III of Madrid UC3M, where the advisors of this Thesis had carried related works.

Not only the Zynq-7000 device family, but also The Zynq Ultrascale+ device family was also considered for this work and presents attractive features such as a more powerful processing system and more advanced programmable logic technology. However, it was discarded because it was a very recently launched product with expensive and scarce available development boards at the date that this Thesis started. Additionally, by that time the radiation effects community had already identified that these devices were susceptible to single-event latchup (SEL) effects [136]–[138], which made them unattractive for space applications. Given the higher availability and maturity of Zynq-7000 family, and its good observed behavior under radiation [139], it was selected instead.

3.1.1.b ZYBO development board

Zynq-7000 devices must be mounted on an electronic board to conveniently support development and testing activities. In this work, commercial ZYBO boards [140], manufactured by Digilent have been extensively used for the development of the error detection and diagnosis techniques, fault injection campaigns, and proton and neutron irradiation campaigns. The ZYBO board mounts the XC7Z010-1CLG400C device of the Xilinx Zynq-7000 family. ZYBO board provides attractive features for application development in a cost-effective product with a compact form factor. The following features have been mainly used in this work:

- Micro SD memory card slot. The board has the capability to boot the Zynq-7000 SoC from a configuration file in this memory. This feature has been extensively used in fault injection and radiation testing.
- PMOD connectors for functionality expansion and connection to other devices. PMOD connectors have been used in testing campaigns to collect data from the device.
- Convenient USB connection for power supply, programming and communications for application development.

- Buttons, LEDs and switches to support application development.

3.1.1.c PicoZed development board

Additionally to the ZYBO boards, a PicoZed development board [141], manufactured by AVNET, was also used to support laser testing campaigns. PicoZed board was mounted on a carrier board, called FMC board to provide connections with the rest of the test setup. The PicoZed 7Z030 was used because it mounts a XC7Z030-1SBG485C device of the Xilinx Zynq-7000 family with the peculiarity that it has not plastic cover so it is very convenient for laser fault injection. Despite the different device part marking, it is functionally equivalent to the one used with the ZYBO boards, as both belong to the Xilinx Zynq-7000 family. PicoZed FMC board provides USB communication, buttons, switches and LEDs for application development and also supports Micro SD system boot and PMOD connectivity. However, the power supply and the programmer are not integrated and must be connected externally.

3.1.2. Facilities

Radiation testing facilities have already been introduced in general terms in Chapter 2. In this section, only the facilities used during the development of this Thesis and its relevant features are discussed.

3.1.2.a Centro Nacional de Aceleradores (CNA), Seville, Spain

The Centro Nacional de Aceleradores (CNA) is a user-oriented accelerator facility in Seville, Spain [142] with various radiation facilities and accelerators. The 18 MeV proton Cyclotron has been extensively used in this work. It is an IBA 18/9 Cyclotron accelerator, capable of accelerating proton up to 18 MeV energy. The cyclotron is installed inside a radiation-shielded room and is remotely controlled by an operator. Several beam lines are installed in this accelerator and most of them are used for radiopharmacy purposes. However, an external line is provided in an adjacent radiation-shielded room to perform experiments in Nuclear Physics. The particle beam passes through a thin window and goes to the air before impacting on the target. This arrangement is very convenient as there is no need to perform vacuum to conduct the experiments and also the space available for the installation of the setup is not limited. A remotely controlled motorized table with step motors allows the positioning of the targets with accuracy of $10\mu m$. The facility provides versatile connections to allow remote control of the experimental setup from the control room. The DMA group at UC3M has a long history of partnership with CNA, pioneering the use of this facility for electronics testing. Fig. 3.1 includes a picture of the setup used at CNA during the tests performed for [J2] in this Thesis. In the picture, the external beam

line of the facility can be observed at the back and the ZYBO board containing the DUT is placed in front of it.

Fig. 3.1. Experimental setup at CNA.

3.1.2.b Los Alamos Neutron Science Center (LANSCE), New Mexico, United States

LANSCE [143] is a National User Facility at Los Alamos National Laboratory (LANL) with one of the most powerful linear accelerators in the United States. It produces energetic protons and neutrons for nuclear physics, material science, national security research and fundamental science. Access to LANSCE radiation facility can be obtained by invitation. The LANSCE experiments reported in this Thesis were performed after submitting a proposal which was accepted by LANSCE's evaluation board. A convenient feature of the neutron facility is that several Devices Under Test (DUTs) can be irradiated at the same time, given the low interaction of neutrons with matter. As a result, different electronic boards with different DUTs can be arranged in line to simultaneously obtain radiation-induced errors in all of them [33], [40]. Fig. 3.2 includes a picture of the setup used at LANSCE during the tests performed for [J5] in this Thesis. In the picture, the aligned arrangement of several DUTs one after the other can be observed.

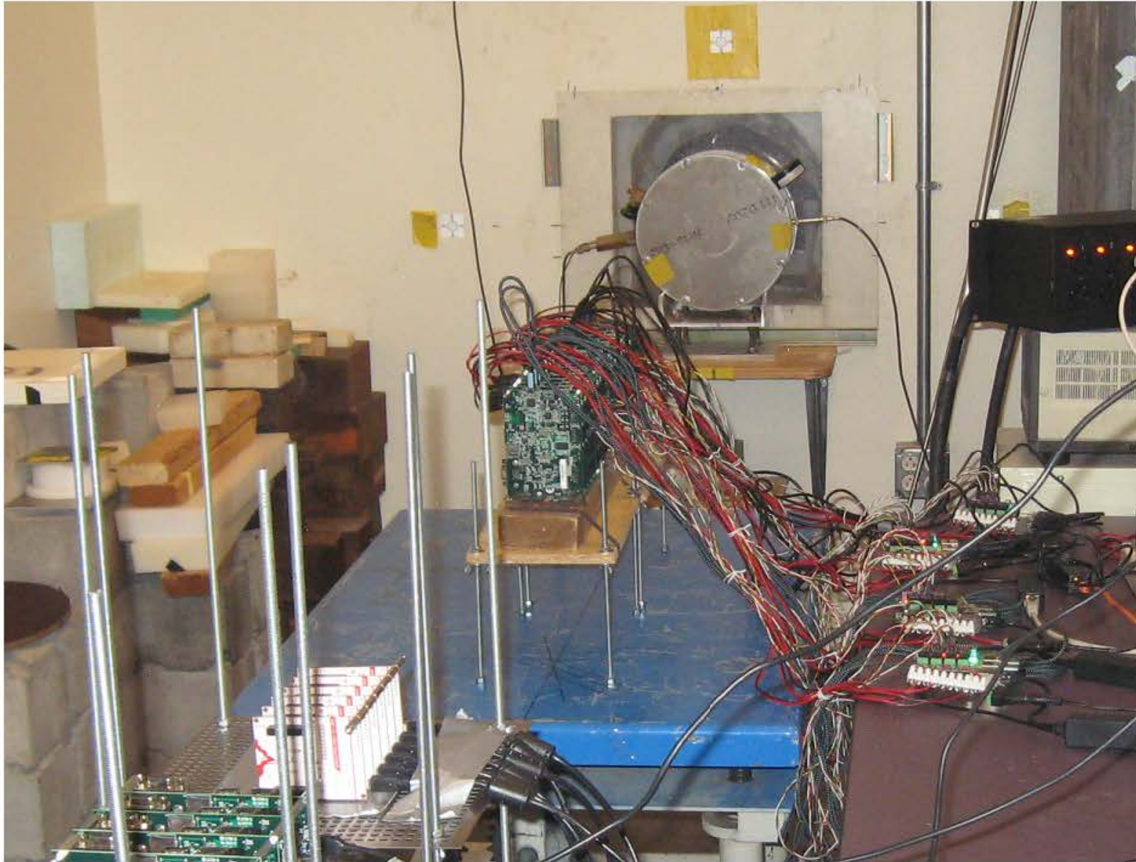

Fig. 3.2. Experimental setup at LANSCE.

3.1.2.c University of Montpellier

The Institut d'Electronique et des Systèmes (IES) of the University of Montpellier has broad experience performing laser fault injection testing. In its facilities, it is possible to perform Single Photon Absorption or Two Photon Absorption injection techniques [42]. Laser injection campaigns accomplished in this work have been performed on the single-photon absorption microscope of IES laser testing facility. The test control equipment allows to configure the laser wavelength, laser energy and the duration of the laser pulses. The beam can be focused down to reduced spot sizes in the micrometer range using magnification lenses. The laser source and optical elements are static and the device is positioned on a motorized stage to allow precise positioning and scanning with micrometer resolution. Additionally the test setup allow to obtain microphotograph images from the surface of the device under test (DUT) to identify determine the areas for fault injection.

The collaboration with University of Montpellier appeared as an synergy opportunity after the presentation of developed error diagnosis techniques at NSREC 2019 conference. Fig. 3.3 includes a picture of the setup used at Montpellier University during the tests performed for [J6] in this work. In the picture, the board is placed on the motorized stage under the laser magnification lenses.

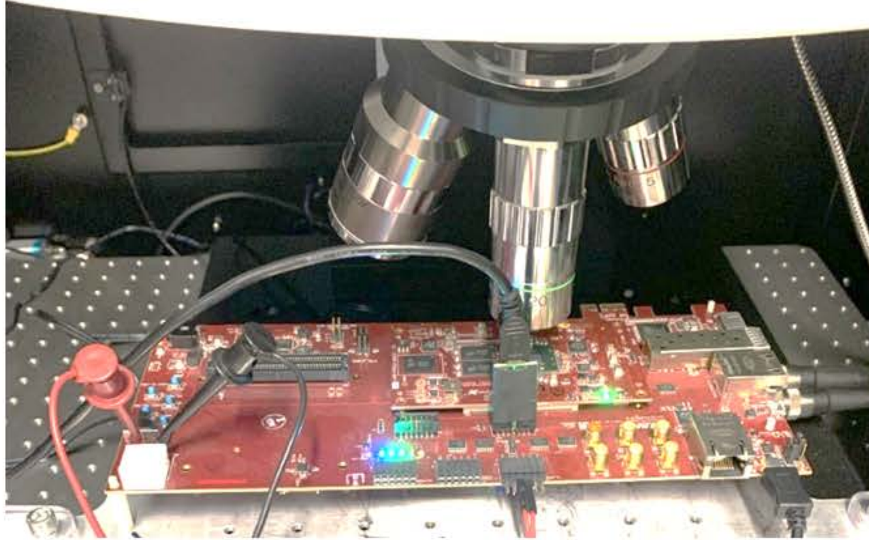

Fig. 3.3. Experimental setup at Montpellier University.

3.1.3. Fault injection tool

This Thesis extensively uses fault injection to conveniently evaluate the developments at desktop level in an inexpensive manner. The DMA group at UC3M has a long experience in fault injection and has contributed key works in this topic. A specific software fault injection methodology was developed for this Thesis based on the Code Emulated Upset (CEU) approach [56]. The developed methodology consist in configuring a timer interrupt in the processing system at a random instant during the execution of the application software. When the interrupt takes place, the content of all processor registers is saved on the stack in the processor memory. During the interrupt, a random bit is changed in a random register in the stack content. When the interrupt is finished, the content of the stack is restored on the processor registers, including the value with the injected error. Alternatively, it is possible to inject an error in a random memory position instead of a register. This mechanism can be easily applied in any type of software, with or without operating system (OS). The described fault injection technique has been developed, with participation of the author, as a code library written in C programming language to be included in any application of interest with low effort.

3.1.4. Other equipment

In addition to the vehicle of study and the access to test facilities, other resources were used during the development of this work.

3.1.4.a Xilinx Vivado and SDK development environment

To develop applications for the Zynq-7000 AP SoC, a programming environment is used. Vivado is the tool provided by Xilinx for FPGA and SoC design and SDK is the software development kit to develop software applications running in the SoC processors.

3.1.4.b Modelsim simulation engine

Modelsim simulator tool provided by Mentor has been used for VHDL simulation. It has been selected due to a convenient command-based interface that supports automation through Python scripting.

3.1.4.c Python scripting language

Python is an interpreted high-level general-purpose programming language with powerful standard libraries and wide open-source community support. It has been used as an scripting language for the automation of data collection task during testing campaigns and also to post-process collected data to extract error counts, statistics and other metrics.

3.1.4.d RaspberryPi

RaspberryPi [144] is a common name for single-board computers (SBCs) developed by the RaspberryPi Foundation that has gained high popularity because of its low cost, modularity, and open design. In this Thesis, RaspberryPi computers have been used to control test setups, oversee the behavior of the devices under test (DUTs) and collect test data during test campaigns. The Department of Electronics Technology of the University Carlos III of Madrid developed a custom test system based on RaspberryPi in combination with electric relays to provide capability to power cycle and control up to four ZYBO boards at the same time. This development was carried out in the frame of this Thesis with the participation of the author. A variation of that development was performed to control the PicoZed board.

3.2. Methodology

Provided that this work has been done under the framework of an Industrial Ph.D. program, the methodology guidance is shared by both the university and the company. Following the applicable regulation, the company designated a tutor to coordinate the activities. As a key organization point, periodic meetings between the academic advisors, the company tutor and the author were carried out. The main points addressed in such meetings were:

- Analysis of partial and final results for each task.
- Revision of schedule and objectives according to the evolution of the activities.
- Look for possible industrial applications of this work.

The development of this work can be divided in four main tasks:

- Development. Control-flow and data error detection and diagnosis techniques have been proposed, developed and implemented on an IP core in VHDL capable of detecting and diagnosing errors using information from the trace interface. This task has been the backbone of the work, and the rest are derived from it.
- Validation. In the radiation effects scientific community it is essential to perform radiation testing for new techniques to be evaluated and the results to be accepted.
- Dissemination. During the development of this work, partial results have been communicated to conferences and published in articles. This Thesis document is the final documentation of the whole work.
- Industrialization. As an Industrial Ph.D. program, the research work has been oriented to produce a result susceptible for industrialization and specific activities have been performed in that way.

Additionally to the main tasks, other activities such as the participation on doctoral meetings, seminars or training courses have been carried out by the author.

In general lines, it can be considered that the activities have been divided between the university and the company as follows:

- University: Proposal, design, development and evaluation of error detection and diagnosis techniques, including radiation testing.
- Company: Look for the application of the developed techniques in other projects of the company and in the space industry.
- Both: Dissemination of results by participation on international conferences and article publications, increasing the visibility of both parties in technical forums related to the topic of this Thesis.

During the development of this Thesis, it was encouraged the contact with other research entities, which was realized through up to three collaborations:

- Collaboration with Centro Nacional de Aceleradores: started in March, 2018, it involved a close interaction to develop a test setup and methodology for this Thesis.

- Collaboration with Alicante University: started in February 2019 with the objective of hardening multicore systems. It allowed the extension of the developed techniques to multicore systems and involved injection and radiation campaigns. This collaboration resulted in two conference communications as oral presentations and two JCR journal articles to date.

- Collaboration with University of Montpellier: started in December 2019 with the objective of aumenting the knowledge on the proposed error diagnosis techniques by the use of laser fault injection. It has allowed deeper understanding of detection and diagnosis capabilities and limitations by means of two laser injection campaigns and has resulted in one conference communication as oral presentation and one published article at IEEE TNS.

Globally, both the university and the company were thoroughly compromised with the development of this Thesis and to closely collaborate in the development of complementary activities to obtain the best possible results from this work.

A the end of each task, outcomes were analyzed and the planning could be updated if needed. Obtained results and associated discussion and conclusions were reflected in a report that could be replaced by a publication with the same purpose.

3.2.1. Chronology

The research work started in January, 2018 and were carried out in four years. In general lines, during the first year, error detection techniques were developed, tested and published. During the second year, the detection capabilities were combined with error diagnosis techniques, resulting in more complex tests and additional publications. Finally the industrialization and the development for this document were done during the the third and fourth year. It is remarkable that the Covid-19 pandemic impacted on the schedule, forcing to move forward some activities of the third year to an additional fourth year. The availability of radiation facilities was also affected by the pandemic, which obliged to modify the planned test campaigns and associated developments.

3.2.2. Development

The developed techniques have been implemented in an IP module designed in VHDL hardware description language. Development activities included:

- Study of the application

- Definition of operation and interfaces

- Trace interface protocol review

- Codification of the IP module in VHDL
- Functional tests
- Design revisions and identification of areas for improvement
- Inclusion of selected improvements in the design
- Documentation of the last version of the design to offer to space industry

The development of the work performed in this Thesis has been a progressive task which is reflected in the evolution of the different published articles included in the compendium in this document. Two linked developments can be distinguished: error detection and error diagnosis techniques.

3.2.2.a Error detection

The error detection techniques were firstly developed addressing control-flow errors. For that purpose, the Program Trace Macrocell (PTM) trace specification [120] was reviewed and implemented in the VHDL IP module. Validation was performed on the first time by VHDL simulation and, after that, the module was implemented in the programmable logic of the Zynq-7000 device on a ZYBO board and debugged against the real trace interface of the processing system.

Hardware debugging was performed by connecting the IP directly to the trace port and checking whether the IP was able to decode and interpret properly the trace information. In the beginning of tests, design mistakes were identified thanks to a dedicated signal in the IP indicating that a decoding error had been produced. In the absence of radiation or any other disturbance, a design mistake was the most probable cause for it. To better analyze the problem, the system was configured to collect the trace data in a buffer and print it at the moment of the error. The collection of the trace at that moment allowed to replicate the same problem on simulation and identify the design mistake. This approach also allowed to obtain high amounts of real trace data to improve the knowledge about the protocol, in a more straightforward and reliable manner than generating artificial trace data manually. With this approach, the IP was quickly debugged.

After the design was stable on the test bench, control flow error detection techniques were tested under fault injection and irradiation campaigns, and published in [J1], becoming a pioneer work in the detection of radiation-induced faults through the trace interface in a hard-core processor.

When the initial implementation of control-flow error detection techniques had been validated, the same methodology was applied to data error detection techniques, which were introduced in the IP by supporting the trace information from the Instrumentation Trace Macrocell (ITM) [118]. VHDL simulations, hardware debugging, trace collection, fault injection and radiation testing activities were also performed. The effective simultaneous

detection of control-flow and data error capabilities of the IP was demonstrated under fault injection and proton irradiation and presented in RADECS 2018. In addition, an extended version of that work was published in [J2]. To the best of our knowledge, this was the first work aiming to detect both data and control-flow errors through the trace interface.

The selection of the software benchmarks running on the processing system during the validation tests is a main key of this process. In the first hardware tests, simple benchmarks were used to deliberately provoke the processor and the trace to generate particular trace packets, for example conditional and unconditional branches of different lengths, or exceptions. However, such custom made benchmarks are not representative of real applications and particular cases may be omitted. For that reason, more complex benchmarks were used later to finish the validation of the design and for fault injection and radiation testing. In this Thesis, the matrix multiplication, quicksort and AES algorithms were selected for the tests as they are very commonly used in this field [52]. The use of common benchmarks also used by other authors in the microprocessor radiation effects community is a remarkable consideration when evaluating and comparing for the effectiveness of different error mitigation techniques [52].

Published results in [J1] and [J2] showed a big potential of the developed error detection techniques both for research purposes, but also as candidates to be integrated in an industrial application. For that reason, it was decided to extend the design activities to enhance IP capabilities. With learned lessons from the first tests, techniques were improved and refined leading to more complex designs. In [J3], a hybrid technique was proposed to leverage the dual-core processing system in the Zynq-7000 device and execute a software application in lockstep with data error detection provided by the replicated execution and control-flow error detection performed by the developed IP simultaneously in both cores. This approach was tested under fault injection showing good error detection capabilities and was later tested under proton irradiation in [J7] also with high detection rates. In [J5], the NEON SIMD coprocessor existing in Zynq-7000 processing system was reused to optimize redundant computations and reduce the overheads of repeated operations in replicated data, while data consistency and control-flow error detection were checked by the developed IP. This work showed a very high error detection rates under proton and neutron irradiation.

As a result, new error detection techniques for control-flow and data errors in microprocessors have been obtained and integrated in a non-intrusive IP that operates online with the processor with low latency.

3.2.2.b Error diagnosis

As an opportunity derived from the development of error detection techniques and particularly due to the collection of trace data for IP design debugging, an error diagnosis approach was envisaged. It turned out that, once the design was debugged, the collected trace data became useful to get a snapshot of the the moments immediately before the error

occurred, both in fault injection and irradiation campaigns. With that information, it was possible to discriminate a wide range of error types and also to contextualize each error in a particular moment in the execution of the application code.

This new feature was first proposed in [J2] to diagnose the cause of radiation-induced errors, which is one of the current main challenges in microprocessor testing and hardening [33]. Diagnosis capabilities would allow to evaluate vulnerabilities of different microprocessor components. Despite error diagnosis was not initially planned, it was found promising and the development was undertaken in parallel with the enhancement of the error detection techniques. Error diagnosis development included the identification of the most relevant trace information to perform error diagnosis and to include in the IP the capabilities to gather such information and provide it in the moment of an error. The culmination of this work was presented in NSREC 2019, gaining high relevance, and an extended version was published in [J4]. After that, an additional evaluation of the diagnosis approach was performed with laser fault injection and published in [J6].

3.2.3. Validation

Developed techniques implemented in the designed IP have been progressively validated throughout the performed activities. Simulation testing was initially performed to validate the implementation of the trace protocol. However, that was only for development purposes. For the radiation effects scientific community to accept new error mitigation techniques, irradiation campaigns must be carried out.

Radiation testing was generally planned to be performed in the first or the last quarter of each year to obtain and analyze data in advance to be communicated to the most relevant conferences in this sector. Conferences generally fix the deadline for communications, so it is important to have the results ready to be communicated in time.

The number of facilities suitable for radiation testing around the globe is limited, and they are highly demanded so the tests must be planned in advance. The main risk of radiation testing is on the availability of radiation facilities and the high cost. To reduce the risk of unavailability, reservation was performed in as much advance as possible and contingency plans were proposed with alternative facilities to minimize the impact in the work schedule.

In this Thesis, six main radiation campaigns were performed: three in CNA with low energy protons, one in LANSCE with neutrons and two in Montpellier University with laser fault injection.

An additional risk of performing radiation testing is to find issues during the test which affect the quality of the obtained data. To maximize success of radiation campaigns, fault injection was performed before each radiation campaign, and using the most similar setup as possible.

The test vehicle for radiation campaigns was mainly the ZYBO board for fault injection, proton and neutron irradiation tests. For these tests, several boards were needed as they became generally damaged by the radiation. Additionally it was necessary to design custom boards or cabling to connect to the vehicle and obtain the information.

The setup used for fault injection and radiation testing was almost similar. The ZYBO test vehicle was equipped with a Micro SD card for automatic boot and connected to a RaspberryPi with a USB cable to provide serial communication. The RaspberryPi also controlled the power supply of the test vehicle to perform a power cycle whenever necessary. Additionally, the RaspberryPi constantly monitored dedicated error signals on the PMOD port of the test vehicle through GPIO to determine whether the IP had detected an error or not.

When performing fault injection tests, the RaspberryPi was used to provide a random number to the application running at the ZYBO board to allow the fault injection to occur in a different moment and location in each execution.

A basic test setup is shown in Fig. 3.4 with a RaspberryPi controlling a single ZYBO board. It was used during fault injection campaigns and at CNA and LANSCE campaigns as can be observed in Fig. 3.1 and Fig. 3.2 respectively, where the RaspberryPi devices are placed next to the DUT, outside the beam.

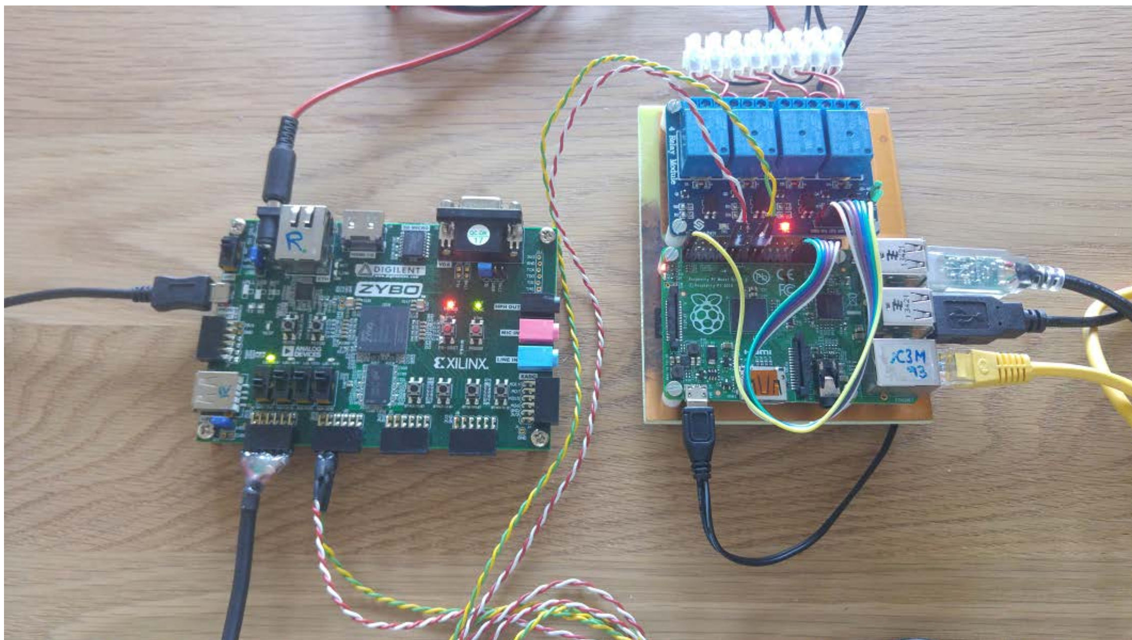

Fig. 3.4. RaspberryPi connected to ZYBO board for test campaigns.

During the tests, the DUT was constantly sending a message to the RaspberryPi indicating its error status according to software checks. The RaspberryPi was executing a Python script to automatically control the experiment and log the messages from the DUT in its internal storage for further analysis. In the case that the RaspberryPi stopped getting the status message, then a SEFI could be assumed. Similarly, in the case that the software-implemented detection techniques on the benchmark detected any error, the status

message was modified to identify the problem. Finally, the state of the dedicated error signals of the IP was also monitored by the RaspberryPi. The RaspberryPi logged any error situation and performed a power cycle on the DUT whenever necessary.

Upon the detection of an error by the IP, the application in the DUT entered in a special function to print the most recent trace data through the serial port. The RaspberryPi collected the trace data and saved it on a dedicated file in its internal storage for further analysis before performing the power cycle.

After the experiments, log files were analyzed and processed with the help of PC-based tools as office programs and custom-developed python scripts for automated generation of results.

3.2.4. Industrialization

The industrialization can be considered as the final step of the development process. However, the whole design was performed with industrial orientation from the beginning. The IP was oriented as a saleable product. By this task, this Thesis wanted to enhance the capabilities and available technologies at Arquimea in the field of microprocessors for space as a competitive advantage. For that purpose, meetings with potential users were carried out to explain industrial applications and gather their interests.

The main task of the industrialization was the proposal of the IP as a product commercialized by Arquimea to be used in space applications. For that purpose, the IP design was documented and refined to obtain a version ready for industrialization. Additionally, potential applications were been identified and proposed. The IP was proposed the European Space Agency (ESA) as a project in an open call for ideas, and it was accepted for contract.

As a result, a new project for the European Space Agency (ESA) involving Arquimea and University Carlos III of Madrid (UC3M) started in the last quarter of 2021. The project, which is out of the scope of this Thesis, has three main objectives:

- To expand the capabilities of the IP developed in this Thesis to support a wider range of ARM devices, by introducing new decoding capabilities for more trace sources.
- To review and document the resulting IP developed to be offered to third-party customers that may be interested in explore opportunities to integrate it in new applications.
- To test the new developments under heavy ion irradiation to obtain new results relevant for the space community to consider the use of the new IP in future missions.

3.2.5. Dissemination

The research performed in this work was communicated to the scientific community by the participation in international conferences and publication in research journals.

It is remarkable that the outcomes of this Thesis were communicated and accepted in four consecutive editions at the most relevant radiation effects conference in Europe, namely RADECS (Radiation and its Effects on Components and Systems). In 2018, a poster communication was presented, and oral presentations were performed by the author in 2019, 2020 and 2021. One communication was also accepted as poster for the most relevant radiation effects conference in America, namely NSREC (Nuclear and Space Radiation Effects Conference). Another relevant European conference in the reliability topic is the European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF), where the author has performed two oral presentations in 2018 and 2019. The developed techniques and test results were presented as they were developed and obtained in a progressive manner.

Given the industrial orientation of this Thesis, the developed work was also presented at the OBDP (On Board Data Processing), industrially-oriented conference organized by ESA in 2021. In this case, the developed IP along with its capabilities was globally presented in an oral presentation as a new building block to be integrated in the processing system of a space application:

[C1] M. Peña-Fernandez, A. Lindoso, and L. Entrena, "IP to detect and diagnose errors in COTS microprocessors through the Trace Interface", *presented at the 2nd European Workshop on On-Board Data Processing (OBDP2021)*, Jun. 2021. DOI: 10.5281/zenodo.5521538

Regarding publications, the most relevant research journal in the radiation effects topic is IEEE Transactions on Nuclear Science (JCR Q2), in which the author published five articles in the framework of this Thesis. Additionally, Microelectronics Reliability by Elsevier (JCR Q3) is another important journal in the reliability field, in which two more articles were published. The complete journal publication list is as follows:

[J1] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, S. Philippe, Y. Morilla, and P. Martin-Holgado, "PTM-based hybrid error-detection architecture for ARM microprocessors", *Microelectronics Reliability*, vol. 88-90, pp. 925–930, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.074 (JCR Q3)

[J2] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martín-Holgado, "Online error detection through trace infrastructure in ARM microprocessors", *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1457–1464, Jul. 2019. DOI: 10.1109/TNS.2019.2921767 (JCR Q2)

[J3] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, M. García-Valderas, L. Entrena, A. Martínez-Álvarez, and S. Cuenca-Asensi, "Dual-core lockstep enhanced with redundant multithread support and control-flow error detection", *Microelectronics Reliability*, vol. 100-101, no. 113447, Sep. 2019. DOI: 10.1016/j.microrel.2019.113447 (JCR Q3)

[J4] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "The use of microprocessor trace infrastructures for radiation-induced fault diagnosis", *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 126–134, Jan. 2020. DOI: 10.1109/TNS.2019.2956204 (JCR Q2)

[J5] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "Error detection and mitigation of data-intensive microprocessor applications using SIMD and trace monitoring", *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1452–1460, Jul. 2020. DOI: 10.1109/TNS.2020.2992299 (JCR Q2)

[J6] M. Peña-Fernandez, A. Lindoso, L. Entrena, I. Lopes, and V. Pouget, "Microprocessor error diagnosis by trace monitoring under laser testing", *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1651–1659, Aug. 2021. DOI: 10.1109/TNS.2021.3067554 (JCR Q2)

[J7] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, S. Cuenca-Asensi, L. Entrena, Y. Morilla, P. Martín-Holgado, and A. Martínez-Álvarez, "Hybrid lockstep technique for soft error mitigation", *IEEE Transactions on Nuclear Science*, 2022. DOI: 10.1109/TNS.2022.3149867 (JCR Q2)

4. PTM-BASED HYBRID ERROR-DETECTION ARCHITECTURE FOR ARM MICROPROCESSORS

Abstract

This work presents a hybrid error detection architecture that uses ARM PTM trace interface to observe ARM microprocessor behaviour. The proposed approach is suitable for COTS microprocessors because it does not modify the microprocessor architecture and is able to detect errors thanks to the reuse of its trace subsystem. Validation has been performed by proton irradiation and fault injection campaigns on a Zynq AP SoC including a Cortex-A9 ARM microprocessor and an implementation of the proposed hardware monitor in programmable logic. Experimental results demonstrate that a high error detection rate can be achieved on a commercial microprocessor.

This chapter has been published as an article.

[J1] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, S. Philippe, Y. Morilla, and P. Martin-Holgado, "PTM-based hybrid error-detection architecture for ARM microprocessors", *Microelectronics Reliability*, vol. 88-90, pp. 925–930, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.074

4.1. Introduction

Microprocessors are commonly used in a wide variety of applications, including safety-critical and high availability missions. In these applications, meeting the reliability requirements in an effective manner is a challenge. Among the multiple factors that may affect reliability, radiation-induced soft errors have the potential to cause the highest failure rate of all other reliability mechanisms combined [27]. Therefore, they are a primary concern in applications working in extreme environments, such as space, and a growing concern also at the ground level.

Although there are radiation-hardened microprocessors specifically developed for these type of environments, they are generally expensive and have high power consumption. Moreover, their performance generally lags behind commercial processors. As a consequence, there is a growing interest in the use of COTS (Commercial Off-The-Shelf) microprocessors even for space applications [6]. In this case, error detection or mitigation must be provided taking into account that the hardware cannot be modified.

Software fault-tolerance techniques [145] introduce redundancy in the code in order to detect or correct errors. These techniques have been widely studied and are the basic solution for COTS microprocessors. However, they are limited because processors contain

many sensitive resources that cannot be directly accessed through software. In addition, they introduce significant performance penalties. These limitations are particularly relevant in the case of control-flow error mitigation [87].

To overcome these limitations, the use of hardware monitoring has been proposed [1]. Hardware monitoring uses an additional piece of hardware that can observe the execution flow of the processor through a suitable interface. Debug resources, which are commonly available in most microprocessors to facilitate system development and software debugging, can be reused for this purpose. These resources are useless during normal operation, so they can be reused for on-line monitoring in an inexpensive way. On the other hand, they can provide internal access to the microprocessor without disturbing it. In particular, the use of program trace interfaces has been proposed and demonstrated for soft cores [1], [5], [122]. In a soft core, it is possible to use a low-level or custom trace interface that provides great flexibility and performance. However, in the case of commercial cores, trace interfaces are usually complex and require trace information to be decoded and synchronized for the application.

In this work we propose and evaluate a hybrid error-detection architecture for ARM processors. ARM is currently one of the most popular choices for embedded systems and supports debug and trace functions through the CoreSight™ subsystem [118]. CoreSight is actually a family of IP (Intellectual Property) modules. In this paper, we focus on the Program Trace Macrocell (PTM), a CoreSight component that provides program-flow trace information. The PTM is the basic program flow trace macrocell for the ARM Cortex-A9 architecture [120].

In the proposed hybrid approach, the code is hardened for data errors, using duplication, while control-flow errors are detected by a hardware monitor attached to the PTM through the trace port. The hardware monitor continuously receives and decodes trace packets along the execution of the application program, extracts the control-flow information and checks it on-line.

Validation of the proposed hybrid architecture has been performed with fault injection and proton irradiation campaigns. Fault injection is a widely used approach to evaluate the effects of faults in an inexpensive way, but it is limited to user accessible components. Additionally, a proton irradiation campaign has been performed to test the proposed hybrid architecture in a more realistic way. Both tests provided very similar results. We show how the combination of data duplication and hardware monitoring provides a good error detection capability. We also evaluate the contribution of each part of the system to the error detection rate.

The remaining of this paper is as follows. Section 4.2 summarizes related work and introduces some concepts about hybrid architectures based on the trace interface. Section 4.3 describes the proposed hybrid architecture. Section 4.4 shows the experimental results. Finally, Section 4.5 presents the conclusions of this work.

4.2. Related work

Microprocessor hardening techniques are usually divided into software, hardware, and hybrid techniques [145]. The type of detected errors by all these techniques is commonly divided into errors affecting control- flow and errors affecting data. Control-flow errors modify the execution flow causing the microcontroller to execute a different instruction than the one that had to be executed. Data errors affect exclusively to program data.

Software techniques modify the application software to detect or correct errors. The main advantages of software techniques are flexibility and ease of implementation. Generally, software techniques require larger execution time and increase memory usage (due to the software modifications and required additional storage for comparison information). Software techniques can be also divided into data and control-flow techniques.

Data techniques are commonly based in duplication. Data duplication consists in duplicating all variables used in a program. Original data and duplicated data must perform the same operations. During program execution, duplicated and original data must be checked. Errors are detected when a difference in both data sets is found. In [65] a set of rules are defined to modify the software for this purpose. This work achieves a very good error coverage but with a high impact in area and execution time. In order to solve these limitations, duplication can be applied at different levels, looking for a trade-off between error coverage and performance penalty. Duplication can be performed at instruction, function or even program level. Other possibilities that are present in the literature to decrease the performance and size penalties are based in reducing the number of data checkpoints or limiting the duplicated data. In [67] instead of duplicating all data, specific variable sets are duplicated. Ref. [66] evaluates the relevance of variables and applies a set of rules for selective duplication in order to reduce the impact of duplication.

The most common software control-flow techniques are based on assertions or signatures. Signature-based techniques commonly divide the program code into basic blocks. A basic block is a set of instructions with no branches except for possibly the last one. At compilation time a signature is assigned to every basic block. At execution time, the signatures are computed and checked at the end of every basic block. It must be noted that compilation time signatures require additional storage that may introduce a significant memory size penalty. Examples of these techniques are CEDA [62], ECCA [58] and YACCA [61]. Assertion- based techniques modify the code by inserting special statements (assertions) that check the data-flow correctness. In this case, error coverage can be affected by the assertion location and also by the information included in it, so that they are application-dependent. An example of the use of assertions can be found in [79].

Hardware techniques modify the circuit architecture to harden it. A well-known example of these techniques is TMR (Triple Modular Redundancy). In the case of COTS microprocessors, the architecture is not commonly available. In addition, a new device

has to be manufactured to include the hardware modifications. These drawbacks make the application of this kind of techniques unfeasible for COTS in most cases.

Alternatively, error detection in microprocessors can be accomplished by connecting additional external hardware modules to observe the system behaviour. The error coverage usually depends on the capacity of observation through the feasible connections. Several works have used this approach, proposing hardware modules [85], [86], [146] that range from simple circuits to very complex ones that could be considered similar in complexity to the observed microprocessor. These hardware modules are commonly named watchdog processors. Watchdog processors can also be classified into active and passive. Passive watchdog processors can be used to check signatures or assertions inserted in the software executed by the microprocessor. They commonly require additional memory to store the values for comparison. Active watchdog processors decrease the memory needs but increase the complexity and the required area. These processors are able to execute a simplified version of the program executed by the microprocessor. Examples of these processors are proposed in [86], [146].

Hybrid techniques combine both software and hardware techniques taking advantage of their individual benefits. The most common approach is to apply software techniques for data-flow hardening, as data is more complex to observe externally, and use the hardware monitor to detect control-flow errors. For instance, in [88] a hardware module is used to monitor the control flow while software fault tolerance techniques are used to detect errors in the data-flow.

Microprocessors are commonly observed through memory buses or through the trace interface. A trace subsystem is commonly included in most microprocessors to support software debugging. When the debugging process is finished, this part of the circuit is not used. In [1], an extensive overview of the use of the trace interface for microprocessor observation is presented. The use of the trace subsystem for on-line monitoring was first proposed in [2] to observe a LEON3 microprocessor. In this work, several microprocessors were executing the same software at different times. During execution, signatures were generated from the available trace information. The coverage can vary depending on the selected information that is used to obtain the signatures. An extended approach was proposed in [3], where critical tasks are replicated (in the same microprocessor or in different microprocessors) and the information provided by the trace interface is compared for both executions. The comparison is accomplished by an external hardware module that computes a signature based on trace information. Other approaches have been proposed that make a more elaborated use of the trace information. In [4], a hybrid technique is proposed using the trace interface to harden the execution flow while data errors are handled with SWIFT-R technique. A new technique was proposed in [5] that compares the program flow information retrieved from two different points: the trace interface and the memory bus. This technique was able to detect all control-flow errors in a LEON3 microprocessor.

4.3. Hybrid architecture

4.3.1. Hardware monitor

In this paper we present a hardware monitor that observes the execution of an ARM Cortex-A9 core through its trace interface. The hardware monitor is capable of decoding and checking program trace information. It has been developed as an IP core that can be configured as a system peripheral. A Xilinx Zynq-7010 [119] All Programmable System-on-Chip (AP SoC) device, including a dual-core ARM Cortex-A9 processing system, has been chosen as the test platform for the proposed system. An overview of the complete system is shown in Fig. 4.1.

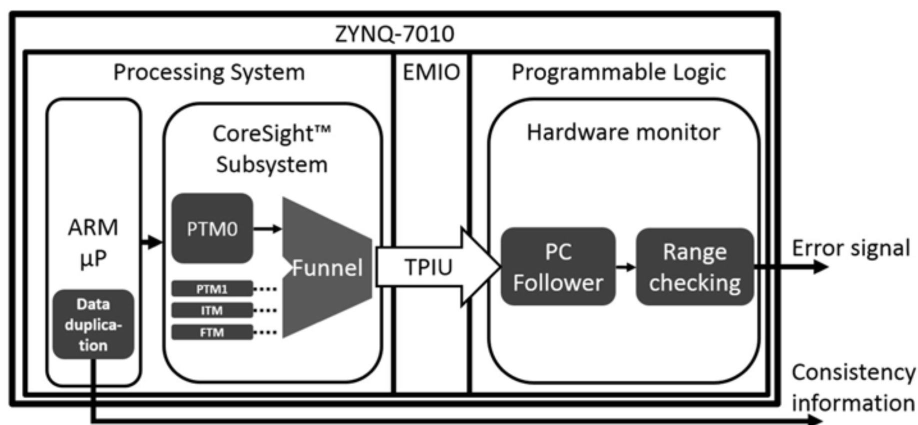

Fig. 4.1. Proposed system overview.

The trace interface provided by the ARM Cortex-A9 is based on the CoreSight™ technology. CoreSight [118] is a family of IP modules intended to support the needs for debug access, instruction tracing, cross-triggering and time-stamping. Some of the most common CoreSight components are represented on the left side of Fig. 4.1 as the Instrumentation Trace Macrocell (ITM), the Fabric Trace Monitor (FTM), the Funnel, or the Trace Port Interface Unit (TPIU). In this work, we focus on one specific CoreSight component, called Program Trace Macrocell (PTM). The PTM is a real-time module that provides instruction tracing of a processor. It is a CoreSight component of the trace source class based on the ARM Program Flow Trace (PFT) architecture specification [120]. Two PTM units are provided in the Zynq-7010 AP SoC, called PTM0 and PTM1, one for each core.

The PTM produces useful information to understand the operation of the processor in a format designed to optimize bandwidth. This is achieved by generating compressed data, which contains just the minimum information required to reconstruct the processor execution flow. To enable correct interpretation of core execution, ARM PFT architecture also provides full information about exceptions, the instruction set state, security state and current Context ID of the processor. The information is formatted in packets. Each packet is composed of a variable, but bounded, number of 8-bit words. To distinguish between

different packet types, the first word of each packet, called the header, must be checked and decoded. The ARM PFT architecture specification ensures a unique header for each packet type to guarantee the correct interpretation of the enclosed information. With respect to this protocol, it is important to note that all packets must be correctly identified and delimited to prevent the monitor from getting lost, regardless of their relevance for the checking process.

A hardware monitor has been developed based on the ARM PFT architecture specification to decode and extract the trace packets generated by the PTM. This monitor has been implemented in the programmable logic of a ZYNQ-7010 and connected to the ARM Processing System through the CoreSight Trace Port Interface Unit (TPIU) using Zynq EMIO (Extended Multiplexed I/O) interface. Trace information is produced by the PTM and driven through the Funnel to the TPIU, so the corresponding Funnel input must be enabled. All involved CoreSight components are configured and enabled by software during the microprocessor initialization.

During operation, the hardware monitor receives and decodes trace packets. In the ARM PFT protocol, the amount of words in each packet is variable and only by identifying the last word in one packet it is possible to identify the header of the next packet. Also, data contained in each word may be relevant to interpret the next ones. For these reasons, a pipelined architecture has been implemented to reliably extract trace information irrespective of the length of the received packets or their order. This way, each packet can be correctly identified and delimited, making the hardware monitor continuously aware of the type of packet which is currently being decoded.

Once the hardware monitor is able to identify and delimit all packet types, any further functionality can be implemented using information available in the received packets. In our application, the available information is used to obtain and monitor the Program Counter (PC) of the ARM processor. The PC value is obtained using information from three main types of packets: I-sync packet, Branch Address packet and Waypoint Update packet. With this method, the PC value can be updated periodically, in every waypoint. ARM PFT architecture specification defines a waypoint as a point where an instruction might involve a change in the program flow. The described functionality is called a PC follower and provides updated PC value information that can then be used to determine the processor behaviour during execution and detect if it is correct or not.

To detect control-flow errors, a range checking method has been implemented. The hardware monitor has been designed to allocate up to eight configurable PC ranges, each of which can be configured through the AXI peripheral interface. These ranges have been named confidence ranges, and they determine allowed PC values during execution. In practice, a user must configure confidence range values with the addresses where the user application functions are stored. Any time the hardware monitor detects that actual PC value is not within any of the valid confidence ranges, an error signal is asserted.

4.3.2. Data error detection

The error detection capabilities provided by the hardware monitor are complemented by conventional software techniques based on data duplication. Basically, all variables are duplicated and all operations are also duplicated on the variable copies. Data consistency checks are also included in the software. They are performed just after variable modification to minimize error detection latency. As mentioned in Section 4.2, variable duplication can be optimized to reduce the overheads. However, in the implementation used in this work all variables were duplicated for the sake of completeness.

4.4. Experimental results

The proposed approach has been tested with proton irradiation and fault injection campaigns. A common experimental setup was used for both experiments in order to make the results as coherent as possible. The experimental setup is described in Section 4.4.1. Then, Sections 4.4.2 and 4.4.3 describe the performed experiments and the results obtained in each case.

4.4.1. Experimental setup

For the experiments we used commercial boards, namely Zybo boards. Zybo contains a XC7Z010 device from Zynq-7000 AP SoC family of Xilinx, which includes a dual core ARM Cortex™-A9 processor. One single core of the device was used, running at the nominal 650 MHz clock frequency. The device also includes a Programmable Logic (PL) part which was used to implement the proposed hardware monitor.

Fig. 4.2 shows a picture of the experimental setup. The Device Under Test (DUT) is included in the Zybo board. The control of the experiment is performed by an external control board that collects and records all the information about the errors that occur during the experiment. The control board can also restart and reset the DUT when non-recoverable errors are observed.

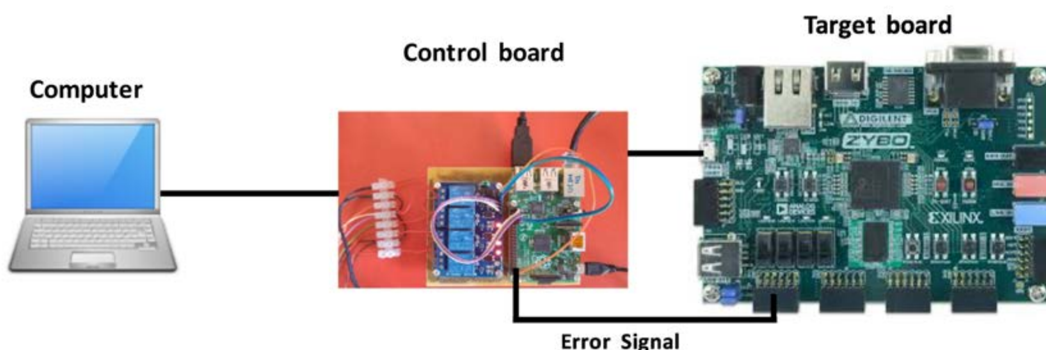

Fig. 4.2. Experimental setup overview.

All the necessary configuration data, including the boot program, the PL configuration bitstream and the application software program are stored in an SD card that is copied to OCM (On Chip Memory) when the device is turned on. As our proposed hardware monitor is located in the programmable logic of the device, it can also be affected by errors. To mitigate these errors, the Xilinx Soft Error Mitigation (SEM) Controller IP was used. Xilinx SEM IP can detect, correct and classify SEUs in the configuration memory of the PL. During the experiments, the SEM is connected to the control board to send all the information about SEU detection, correction and classification. Errors in programmable logic that cannot be corrected by the SEM trigger a reprogramming of the device by the external control board.

We have used a matrix multiplication application software benchmark for the experiments. Following the proposed hybrid solution, the software benchmark was modified to implement data duplication, as described in Section 4.3.2. The benchmark was compiled with Xilinx SDK environment and minimum optimization effort (-O0) in order to prevent the compiler from eliminating duplicated variables. The benchmark is running an infinite loop computing the matrix multiplication of 32×32 elements arrays. The complete software size is approximately 111 kB.

Errors were classified according to the following categories:

- Hang error: Microprocessor cannot continue normal execution and requires the system to be restarted and reconfigured. This category takes into account Hang errors that are detected only by the external control board.
- Detected Hang error (Det Hang): Hang errors that are detected by the proposed hardware monitor and the external control board.
- Detected data error (Det. SW): Software data duplication has detected a discrepancy in duplicated data. These data errors are detected by the software checks and reported to the external control board.
- Communication error (Comm): Communication between the FPGA and the external control board is experiencing a malfunctioning.

4.4.2. Proton irradiation

In order to test the proposed approach we have performed a proton irradiation experiment that took place in March of 2018 at CNA (Centro Nacional de Aceleradores), Spain. The experiments were performed using the external beam line installed in the 18/9 IBA compact cyclotron. The DUT was irradiated in open air with 15 MeV protons. The energy of incident protons in the silicon active area is in the order of 10 MeV, which is enough to produce events for the used technology of 28 nm without the need for thinning the devices [147]. The total fluence was $1.6 \times 10^{12} p/cm^2$.

Table 4.1 shows the number of observed errors and their percentage with respect to the total number of errors for each error category.

Table 4.1. EXPERIMENTAL RESULTS OF PROTON IRRADIATION.

Error type	#Errors	%Errors
Hang	7	2.30%
Comm	3	0.98%
Det. SW	236	77.38%
Det Hang	59	19.34%
Total	305	100.00%

Experimental results show that the proposed approach presents a high capacity of error detection and is able to detect 96.72% of the observed errors. Considering the errors that can be observed by the external hardware monitor, it can detect 89.39% of the observed Hang errors.

The total cross section is $1.91 \times 10^{-10} cm^2$ with a 95% confidence interval between $1.69 \times 10^{-10} cm^2$ and $2.12 \times 10^{-10} cm^2$. When only undetected errors are considered (categories Comm and Hang from Table 4.1), the cross-section reduces to $6.25 \times 10^{-12} cm^2$ ($3.0 \times 10^{-12} cm^2$ - $1.15 \times 10^{-11} cm^2$), which is more than 30 times smaller.

4.4.3. Fault injection

Complementarily to the proton irradiation experiment, we tested our proposed approach with fault injection. We injected faults in the registers in the microprocessor to evaluate the microprocessor behaviour in a more detailed way. For the injection we have used the very same experimental setup that was utilized in the radiation experiments.

The implemented fault injection approach is based on the Code Emulation Upset technique [56]. This approach is summarized as follows. A timer is configured to trigger an interrupt at a random instant. Upon interruption, the full set of registers of the ARM microprocessor is saved on the stack, so that they are available for fault injection. Then, a bit-flip is injected in a randomly selected bit of one of the registers. When the processor returns from the interrupt, the registers are restored from the stack and the single bit injected fault becomes effective. The execution is resumed and is let running for several iterations of the tested application software.

A preliminary run of the application software is used to measure its execution time, which is used as the maximum range for random generation of injection instants. The injected register and bit are also randomly selected. Random seeds are generated externally and provided to the device when it is restarted in order to ensure that random values are generated without bias.

The ARM processor contains a large set of registers. In particular, it uses banked copies of some registers, with the current register selected by the execution mode. In addition, the Single-Instruction Multiple Data (SIMD) and floating-point coprocessors have their own set of registers, which can also be saved in the stack. Faults can be injected in any register using our approach. However, fault injection was performed only on the ARM core registers at the application level view to avoid unnecessarily injecting faults in registers which were not used. Some registers need to be treated in a specific way considering their behaviour. Especially, fault injection in the Program Counter (PC) was actually implemented through the Link Register (LR), because the PC takes the contents of the LR upon return from interrupt.

The results of the fault injection campaign are summarized in Table 4.2. We injected a total of 53,488 faults, of which 12,040 (23.46%) produced observable errors. The proposed approach detected 95.94% of the errors with a 95% confidence interval of ±1.71%. The external monitor detected 89.65% of the Hang errors, with a 95% confidence interval of ±2.65%. These results are in line with those obtained in the proton irradiation experiment, which are within the calculated 95% confidence intervals. The only significant difference is that Hang errors occurred more frequently under fault injection, which is due to the narrower focus of the fault injection experiment. However, the hardware monitor was able to detect a similar amount of errors in both experiments.

Table 4.2. EXPERIMENTAL RESULTS OF FAULT INJECTION.

Error type	#Errors	%Errors
Hang	509	4.06%
Comm	0	0.00%
Det. SW	7631	60.81%
Det Hang	4409	35.13%
Total	12549	100.00%

Fig. 4.3 shows a comparison of errors on a register basis. Fault injection was performed in the complete register file, but only a subset of registers was really used due to the compilation options. For clarity, only the registers that produce at least one error are reported. The frame pointer (FP) and the program counter (PC) are the most critical registers and provoke an error for almost every bit-flip injected in them. For the rest of the registers, error sensitivity may vary depending on their usage.

4.5. Conclusions and future work

This work presents a hybrid architecture that can monitor ARM microprocessor execution thanks to the observation of the program flow trace provided by PTM trace module. This solution presents a feasible and non-intrusive way of detecting errors in

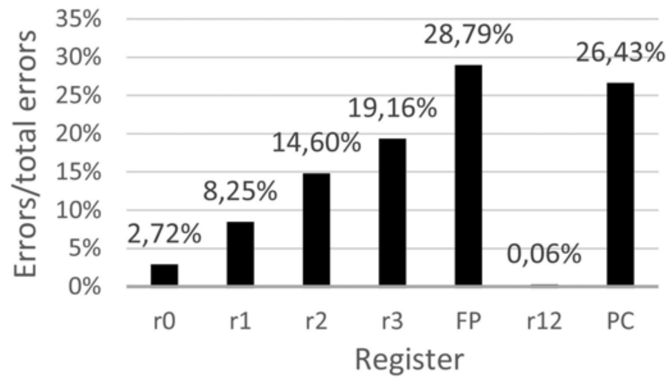

Fig. 4.3. Errors by register.

ARM-based COTS with reduced impact in area. Experimental results demonstrate the high error detection capabilities of the proposed approach.

The proposed approach has been tested with proton irradiation and fault injection. Notably, the results of both tests were very similar, although fault injection was limited to the ARM core registers. Future work is oriented to enhance the error detection capabilities based on the trace information.

Acknowledgements

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under project ESP2015-68245-C4-1-P and by the Community of Madrid under grant IND2017/TIC-7776.

References

[1] 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, pp. 293–306.

[2] M. Grosso, M. S. Reorda, M. Portela-Garcia, M. García-Valderas, C. López-Ongil, and L. Entrena, “An on-line fault detection technique based on embedded debug features”, *Proc. 16th IEEE International On-Line Testing Symposium*, pp. 167–172, 2010.

[3] M. Portela-García *et al.*, “On the use of embedded debug features for permanent and transient fault resilience in microprocessors”, *Microprocessors and Microsystems*, vol. 36, no. 5, pp. 334–343, 2012.

[4] L. Parra *et al.*, “Efficient mitigation of data and control flow errors in microprocessors”, *IEEE Transactions on Nuclear Science*, vol. 61, no. 4, pp. 1590–1596, 2014.

[5] L. Parra *et al.*, “A new hybrid nonintrusive error-detection technique using dual control-flow monitoring”, *IEEE Transactions on Nuclear Science*, vol. 61, no. 6, pp. 3236–3243, 2014.

[6] L. Entrena *et al.*, “Flexible approaches to fault-tolerant microprocessors for space applications”, *Proc. Data Systems in Aerospace (DASIA), ESA Special Publication SP-732*, May 2015.

[27] R. C. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies”, *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[56] R. Velazco, S. Rezgui, and R. Ecoffet, “Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulating Upsets) injection”, *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2405–2411, Dec. 2000.

[58] V. Nair, H. Kim, N. Krishnamurthy, and J. Abraham, “Design and evaluation of automated high-level checks for signal processing applications”, *Proc. SPIE advanced algorithms and architectures for signal processing conference*, Aug. 1996.

[61] O. Goloubeva, M. Rebaudengo, M. Sonza Reorda, and M. Violante, “Soft-error detection using control flow assertions”, *Proc. 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 581–588, 2003.

[62] R. Vemu and J. A. Abraham, “CEDA: Control-flow error detection through assertions”, *Proc. 12th IEEE International On-Line Testing Symposium (IOLTS)*, pp. 151–158, 2006.

[65] P. Cheynet, B. Nicolescu, R. Velazco, M. Rebaudengo, M. Sonza Reorda, and M. Violante, “Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors”, *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2231–2236, 2000.

[66] B. Nicolescu and R. Velazco, “Detecting soft errors by a purely software approach: Method, tools and experimental results”, *Design, Automation and Test in Europe Conference*, pp. 57–62, 2003.

[67] A. Benso, S. Chiusano, P. Prinetto, and L. Tagliaferri, “A C/C++ source-to-source compiler for dependable applications”, *IEEE International Conference on Dependable Systems and Networks*, pp. 71–78, 2000.

[79] M. Hiller, “Executable assertions for detecting data errors in embedded control systems”, *Proc. International Conference on Dependable Systems and Networks*, pp. 24–33, 2000.

[85] A. Benso, S. Di Carlo, G. Di Natale, and P. Prinetto, “A watchdog processor to detect data and control flow errors”, in *9th IEEE On-Line Testing Symposium*, 2003, pp. 144–148.

[86] S. Bergaoui and R. Leveugle, “IDSM: An improved control flow checking approach with disjoint signature monitoring”, *roc. 24th Conference on Design of Circuits and Integrated Systems (DCIS)*, 2009.

[87] J. Azambuja, S. Pagliarini, L. Rosa, and F. Kastensmidt, “Exploring the limitations of software-only techniques in see detection coverage”, *Journal of Electronic Testing*, vol. 27, no. 4, pp. 541–550, 2011.

[88] J. R. Azambuja, M. Altieri, J. Becker, and F. L. Kastensmidt, “HETA: Hybrid error-detection technique using assertions”, *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2805–2812, 2013.

[118] ARM Inc, *CoreSight Components – Technical Reference Manual*. 2009.

[119] Xilinx Inc, *Zynq-7000 All Programmable SoC: Technical Reference Manual, UG585*. 2016.

[120] ARM Inc, *CoreSight Program Flow Trace Architecture Specification*. 2011.

[122] B. Du, E. Sanchez, M. S. Reorda, J. P. Acle, and A. Tsertov, “FPGA-controlled PCBA power-on self-test using processor’s debug features”, *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2016.

[145] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, 2011.

[146] T. Michel, R. Leveugle, and G. Saucier, “A new approach to control flow checking without program modification”, *Proc. 21th International Symposium on Fault-Tolerant Computing (FTCS-21)*, pp. 334–341, 1991.

[147] A. Lindoso, M. García-Valderas, L. Entrena, Y. Morilla, and P. Martín-Holgado, “Evaluation of the suitability of NEON SIMD microprocessor extensions under proton irradiation”, *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1835–1842, Aug. 2018.

5. ONLINE ERROR DETECTION THROUGH TRACE INFRASTRUCTURE IN ARM MICROPROCESSORS

Abstract

This paper presents a solution for error detection in ARM microprocessors based on the use of the trace infrastructure. This approach uses the Program and Instrumentation Trace Macrocells that are part of ARM's CoreSight architecture to detect control-flow and data-flow errors, respectively. The proposed approach has been tested with low-energy protons. Experimental results demonstrate high accuracy with up to 95% of observed errors detected in a commercial microprocessor with no hardware modification. In addition, it is shown how the proposed approach can be useful for further analysis and diagnosis of the cause of errors.

This chapter has been published as an article.

[J2] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martín-Holgado, "Online error detection through trace infrastructure in ARM microprocessors", *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1457–1464, Jul. 2019. DOI: 10.1109/TNS.2019.2921767

URI: http://hdl.handle.net/10016/32697

6. DUAL-CORE LOCKSTEP ENHANCED WITH REDUNDANT MULTITHREAD SUPPORT AND CONTROL-FLOW ERROR DETECTION

Abstract

This work presents a new Dual-Core LockStep approach to enhance fault tolerance in microprocessors. The proposed technique is based on the combination of software-based data checking and trace-based control-flow checking through an external hardware module. The hardware module is connected to the trace interface and is able to observe the execution of all the processors in the architecture. The proposed approach has been implemented for a dual core commercial processor. Experimental results demonstrate that the proposed technique has a high error detection capability with up to 99.63% error coverage.

This chapter has been published as an article.

[J3] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, M. García-Valderas, L. Entrena, A. Martínez-Álvarez, and S. Cuenca-Asensi, "Dual-core lockstep enhanced with redundant multithread support and control-flow error detection", *Microelectronics Reliability*, vol. 100-101, no. 113447, Sep. 2019. DOI: 10.1016/j.microrel.2019.113447

6.1. Introduction

Microprocessors are the backbone of digital electronic systems. The progress of manufacturing technologies and the reduction of the transistor feature size have made microprocessors cheap and suitable for a huge variety of applications. At the same time, the susceptibility to soft errors, mainly caused by ionizing particles, has grown to become a concern [27] in an increasing number of cases. For high reliability applications, microprocessors are required to be fault-tolerant, i.e., to be able to continue operation in the event of failure. In the past, fault-tolerant microprocessors were required for systems working in harsh environments, such as aerospace, but today they are increasingly demanded even at ground level.

Techniques to protect microprocessors against soft errors can be classified into software and hardware techniques. Software techniques are very flexible, but they are inherently limited [87]. Hardware techniques that require modifying the microprocessor are often not feasible because the design and manufacturing of a microprocessor is a costly process that can only be afforded for high volume production. In contrast, Dual Modular Redundancy (DMR) is an attractive solution with high error detection capabilities. As microprocessors

are today rather cheap, duplication is not expensive. In fact, multicore devices have become very common even for low-end devices, and state-of-the art Micro Controller Units (MCUs) are starting to introduce safety features, which are becoming more relevant to automatic control, such as in the autonomous automotive industry and aerospace [154].

Dual-Core Lockstep (DCLS) [82], [155]–[157] is a DMR fault-tolerant technique that can exploit the availability of multicore devices. It consists in two processors simultaneously running the same set of operations, syncing their output each cycle and triggering a recovery routine in case of discrepancy. This architecture is described in the white paper ISO 26262, where the DCLS processors are also referred as "ASIL-D MCUs". Despite the safety features introduced, ASIL-D MCUs do not eliminate the need to implement other safety measures at software and system level. Several MCU's and processors have successfully implemented this feature, for instance, Freescale MPC5643L [158], PPC405 Lockstep System on ML310 and the ARM Cortex-M33, Cortex-R4, Cortex-R5 and Cortex-R7 [83], [154]. The ARM Cortex-R5, has been integrated in several platforms, such as TI Hercules TMS570 microcontrollers and in the Xilinx UltraScale MPSoCs. However, it has been reported that the recovery process presents high overheads, around x1000 compared to a Triple Core Lock-Step [83].

Software DCLS divides the processing into steps, ranging from individual instructions to a set of functions. After each step, the results of the computations produced by each processor are compared. If they do not match, a rollback mechanism is triggered to restore the system back to a consistent state. The comparison of the computation results provided by the two processors is the key aspect of DCLS. Generally, only output data are checked for errors [82], [156]. However, control-flow errors may cause one of the processors to lose synchronization and eventually hang or get lost. Control-flow errors are not easy to detect as they may not have an immediate observable effect in the computed data. Moreover, it is common in dual cores that one of the processors acts as a master and the other as a slave. In such a case, the hang of the master can lead to the crash of the entire system. A possible solution to this problem is to use timeout watchdog monitors to detect unusually long computation times [82]. However, this approach is weak and results in a high error detection latency. Moreover, control-flow errors may produce latent effects that may remain in the system after it is restored even though the output data are correct.

In this work we propose an enhanced DCLS approach that uses two complementary mechanisms: observation of information provided by the trace subsystem to monitor the execution control flow and a multithread software-based scheme to detect and recover from data inconsistencies.

Most microprocessors today provide a trace subsystem for debugging purposes which is able to report the microprocessor control flow in a seamless and non-intrusive manner without affecting the execution. Under normal operation, the trace subsystem can be reused to monitor the control flow of the processor [1], [3]. Errors that affect the control flow

in any of the processors can be detected by on-line decoding the corresponding program traces and checking the obtained information [J1].

Modern processor architectures and Operating Systems (OS) commonly support the parallel execution of different threads and processes. Those capabilities have been exploited in different approaches by executing several replicas of the code on the same processor (SMT-Simultaneous Multithread) [74], on separate cores (CMP-Chip Level Multiprocessor) [76] or using a mixture of them [77]. All those approaches rely on complex software stacks that include, in addition to the OS, different support libraries in order to reduce the development time and ease the management of the replicated threads/processes [159]–[161]. However, every software layer added introduces new vulnerabilities that degrade the overall reliability of the applications.

In our approach, a CMP scheme has been adopted for bare metal applications (without OS) which renders a reduced number of race conditions and lower control overhead compared to traditional solutions. The redundant threads execute on different cores and, eventually, check their outputs. In case of discrepancies, threads are forced to re-execute the critical regions as recovery mechanism. The proposed approach can be considered a relaxed lockstep execution where protection may be applied with different granularity, from the whole application to just some critical regions of the code. Therefore, a suitable trade off can be established between the number checkpoints and the time overhead produced in case of recovery.

The proposed approach has been implemented and evaluated on a dual-core ARM Cortex-A9 [162]. The microprocessor is a hard core in a Zynq FPGA [119]. The proposed trace monitor has been implemented in the programmable logic. The proposed technique has been tested with an injection campaign of 871,837 faults that resulted in 43,769 errors. Experimental results show that the proposed approach shows excellent error detection capabilities with a percentage of detected errors of up to 99.63%.

This paper is organized as follows: Section 6.2 describes the proposed lockstep approach, Section 6.3 presents the experimental results and finally Section 6.4 summarizes the conclusion of this work.

6.2. Proposed lockstep approach

6.2.1. Architecture

Contrarily to other approaches that propose new hardware structures to extend the architecture of CMP processors, our solution is intended to be directly applied to modern multicore processors. The architecture uses redundant multithread support for data error detection combined with trace monitoring for control-flow error detection.

Fig. 6.1 shows the architecture of the proposed DCLS approach. It is divided in three main blocks: multicore microprocessor (Multicore), ARM Trace subsystem (Coresight)

and Control Flow Monitor. The Control-Flow Monitor is a small piece of hardware that can be embedded in a FPGA. Data error detection is implemented in the Multicore block and is described in Subsection 2.2. The remaining blocks (Coresight and Control-flow monitor) are described in Subsection 2.3.

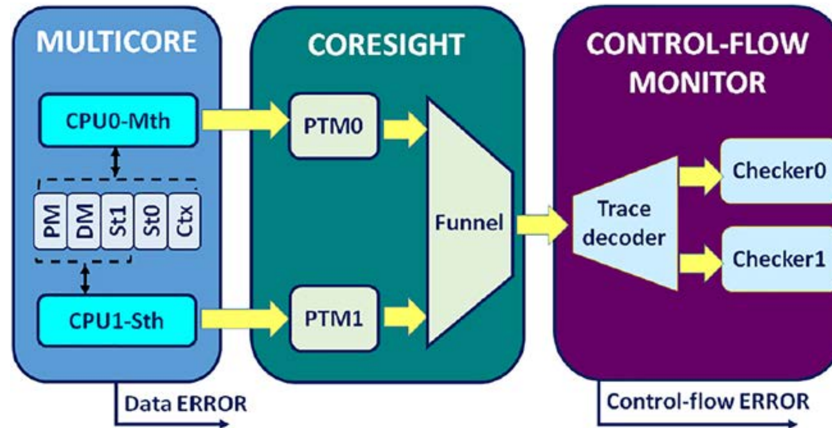

Fig. 6.1. Dual-Core LockStep architecture.

6.2.2. Data error detection

A pure software mechanism has been developed for data error detecting and recovering. It is based on modular redundancy strategy and relies on the parallel execution of redundant threads on separate cores. This way it could be easily adapted to Dual or Triple Modular Redundancy depending on the number of available cores [163].

Similar to traditional threads supported by OS like Linux (POSIX Threads API) or specialized libraries (OpenMP API), in our model the two threads, named master thread (Mth) and shadow thread (Sth) respectively, share instructions (see PM on Fig. 6.1) and data memories (see DM on Fig. 6.1). However, as a key difference, each thread has its own stack (i.e. the memory region for storing temporary and automatic variables and function output variables) which allows holding the replicas of the data. In addition, a third stack memory region is reserved to store the Context (Fig. 6.1: Ctx). Since Mth is in charge of checking the Context and output variables of every region under consideration, this thread has access to the three stacks.

The application is divided into several Critical Regions of code. Each one is characterized by the input or Context variables (i.e. global and local variables) and output variables. For the sake of completeness, in this work the Context contains all the information needed to define the status of the execution, i.e. the input variables to the region. As a limitation of the current implementation of our approach, we are not able to save the status of the processor's cache, and therefore we assume it is disabled. As a side effect of this choice, the application should be written in such a way that input variables are not modified during the execution of the code within a region, so that in case of recovery, the integrity of those variables is preserved.

Critical regions are delimited by annotation primitives in C or C++ and follow the concept of Sphere of Replication (SoR) established in [73]. To this end, the code section is instrumented to check the correctness of the variables and to synchronize the execution using barriers and mutex (mutual exclusion). The SoR defines the code regions where thread replication and parallel execution will take place. When the instruction flow reaches the SoR (critical region), a Context Check is performed by the Mth. If there are no discrepancies, the Context variables are saved to the Context Stack by the Mth. Every time the instruction flow goes out the SoR boundaries, consistency checks are automatically carried out by the Mth on the data stored on both stacks. A recovery procedure, i.e. the re-execution of the critical region, is executed if any discrepancy is found.

Depending on the boundaries and the situation of the critical regions within the code, the protection can be applied at different levels of granularity to get the best trade-off between performance overhead and latency to recover from a fault.

Fig. 6.2 shows an example of annotated pseudocode for the Matrix Multiplication algorithm. As can be seen, SYNC and CHECK annotations enclose the inner-most loop and define a region. Also, this mechanism provides synchronization for each thread and automatic check over Context variables i,j and output variable acc respectively. The localization of the region defines a lockstep execution with NxN checkpoints (steps) and the recovery latency is equivalent to the execution time of the inner loop (lines 9–13) plus an additional assignment (line 9). A finer granularity could be obtained by including just the inner loop body in the critical region. As a result, the recovery latency decreases to just the execution of a line of code, at the cost of increasing the number of checkpoints up to NxNxN. Note that by removing the 8th and 12th lines (which are responsible for the automatic code instrumentation) we obtain the original unprotected code. It is remarkable that both SYNC and CHECK functions can be compiled along with the software application as they are written in C. Thus, to protect an application, the user is just required to manually introduce both functions enclosing the critical sections in the source code. In the future, a tool to insert automatically the calls for these functions is planned to be developed to enhance the scalability of this solution.

In addition to the code annotation, other software tweaks were implemented to endow the dual core system with the ability of running redundant threads on bare metal. In first place, it was necessary to modify the Board Support Package (BSP) in order to initialize the platform and start up all cores presented in the architecture. It is common that, in bare metal environments, the BSP provides the minimal files to boot up the platform. However, BSPs only cover a little subset of the most common ways to boot a system. The default boot sequence is controlled by CPU0 while the other CPU gets into an infinite busy-waiting loop. This default initialization code was changed to allow a boot in SPMD mode (Single Program Multiple Data). In second place, the memory map and the associated linker scripts were modified to support separate stack sections for each core. Finally, a spin-lock mechanism was added to allow the synchronization of the cores.

Algorithm 1:

```
1: NT= 2      //Number of threads
2: A = Matrix[N][N]
3: B = Matrix[N][N]
4: C = Matrix[N][N]
5: procedure MxM
6:   for i = 0 to N-1 do
7:     for j = 0 to N-1 do
         //Start of region
8:       SYNC(i,j;NT)
9:       acc= 0
10:      for k=0 to N-1 do
11:        acc += A[i][k]· B[k][j]
12:      CHECK(acc;NT)
         //End of region
13:      C[i][j]=acc
```

Fig. 6.2. Redundant threaded matrix multiplication.

6.2.3. Control-flow error detection

The control flow of both cores is observed through an external hardware IP (Control-flow Monitor), which is connected to the trace interface. The proposed approach is a multicore extension of the technique presented in [J1].

The dual-core ARM Cortex-A9 presented in the selected device contains a trace subsystem based in CoreSight modules [118]. Of the available CoreSight modules in the selected device, our system uses only the PTM (Program Trace Macrocell) [120]. PTM is a trace source CoreSight subtype. A PTM cell is associated to a single core of the architecture. Thus, for this work two PTM instances, PTM0 and PTM1, are used, which are linked respectively to core 0 and core 1 (Fig. 6.1). Both trace sources are multiplexed and sent through the trace interface. The external IP decodes the trace information and checks the correctness of the execution-flow by controlling the PC addresses of the executed instructions in both cores.

Two techniques are used to detect incorrect execution: confidence range checking and address watchdog checking. The former consists in configuring the external IP to treat some instruction-memory address ranges as valid. While the instruction addresses of a core are within its own confidence ranges, no error is assumed. The latter is also related with the core instruction address, but in this case only one specific address is configured to be checked periodically, namely the first instruction of each step. It the required instruction address is not received within a configurable time, a timeout is asserted. In the case an unexpected instruction or timeout is detected in any of the two cores, an error signal is triggered. The IP works on-line with very small latency. No additional information is required or stored before execution takes place, and the required configuration register values can be determined at compilation time. The external IP can be configured from software as an AXI peripheral.

To ease the use of the external IP, all user-defined application functions have been targeted to a specific region of memory defined by the programmer in the linker script, so most of the code is inside the same confidence interval. Some native or library functions that cannot be targeted to this region have also been protected using three more confidence intervals. The first instruction of the main loop of the code has been selected as the instruction address to be checked by the watchdog. As the application is hardened with DCLS, the external IP has been configured to check both CPUs with the very same parameters.

6.3. Experimental results

A fault injection campaign has been performed to test the proposed technique. Faults were injected only in one of the two cores in the selected architecture. The Mth core have been selected for injection as it is the most critical considering that it performs context and data checking.

Faults were injected in the register file adapting the technique presented in [J1]. This technique generates bit-flips randomly in the register file. An external controller has been used in order to determine, classify and collect the observed and detected errors. In radiation environments, errors can also affect memories which are not covered by the utilized technique. Memories that are exposed to radiation are usually protected with redundancy techniques such as EDAC. In previous radiation campaigns [J1], we have validated the fault injection approach with quite accurate error detection match between radiation and injection results even though we only injected faults in the register file.

The control-flow monitor is located in the FPGA and radiation can affect its behaviour. Xilinx SEM IP [152] can be used to protect the FPGA and the circuit it contains.

The experiments have been carried out on commercial ZYBO boards featuring a Xilinx Z7010 Zynq [119] as the device under test (DUT). Both ARM Cortex-A9 cores in the DUT are clocked at 650 MHz frequency. At the beginning of the application, the external controller generates a random seed which is used to generate the injection parameters (time instant, register number and bit index) and the initialization values of program data. When the DUT has received the seed, the execution starts, and the injector as well. A fault is injected every five iterations of the application main loop. In the case no error appears in these five iterations, a silent error is assumed, and a new injection is produced. In the event of an error, the external controller registers the results and power cycles the DUT to start a new injection.

The results of the fault injection campaign are summarized in Table 6.1. Two experiments were accomplished with two versions of a matrix multiplication benchmark: unoptimized (-O0: column 2 of Table 6.1) and optimized with maximum effort (-O3, column 3 of Table 6.1). Both benchmarks use matrices of 32×32 32-bit integer elements.

Table 6.1. INJECTION CAMPAIGN RESULTS.

Error type	-O0		-O3	
	# errors	% errors	# errors	% errors
Det. Hang	15,462	77.27%	15,879	66.84%
Hang	23	0.11%	64	0.27%
Det. Only IP	75	0.37%	135	0.57%
Det. Data	4338	21.68%	6725	28.31%
SDC	50	0.25%	940	3.96%
Comm	63	0.31%	15	0.06%
Total	20011	100.00%	23758	100.00%

For -O0 benchmark 591,821 faults were injected resulting in 20,011 (3.38%) errors. For -O3 benchmark 280,016 faults were injected resulting in 23,758 (8.48%) errors.

The error categories reported in Table 6.1 are:

- Det. Hang: The fault has produced a functional interrupt in the system, which has been detected by the external IP.
- Hang: The fault has produced a functional interrupt in the system, which has not been detected.
- Det. Only IP: The external IP has reported a control-flow error while no functional interrupt has been produced.
- Det. Data: The fault has produced a data error which has been detected by the software.
- SDC: Silent Data Corruption, the fault has produced a data error which has not been detected.
- Comm: Communications malfunction between the DUT and the external controller that makes impossible to classify the error.
- Total: Total number of errors

Table 6.1 shows the high error detection capability of the proposed approach, with 99.63% errors detected for the unoptimized version and 95.77% for the optimized one. For this metric we have not considered Comm errors, as it is not possible to categorize them.

Regarding control-flow errors, it is noticeable that most of them are related with system functional interrupts, which are mainly caused by exceptions or loss of lockstep synchronization. However, few of them (Det. Only IP) have not produced this effect.

These can be associated with control-flow errors that do not produce a hang on the system, for example, an error that causes a branch to a wrong, but valid, code region or an error in a loop index causing an unexpectedly bigger execution time. Although these errors can be false positives, it is highly recommendable to consider them as real errors for preventive reasons. Code optimization produces a small reduction on control-flow error detection rates.

With respect to data errors, there is a small portion of SDCs. These errors are caused by faults that are injected after the software check. Note that fault injection is non-stop, so data may be corrupted at any time and therefore an error may appear at the final check of the test used to categorize the results. Nevertheless, such errors could be detected if the final check is also made in lockstep. Anyhow, SDC errors represent a very small portion, particularly when executing unoptimized code. This effect increases when optimization is introduced because the compiler changes the order in which some operations are made to achieve higher throughput. Also, as the optimized code gets shorter, the probability to inject an error after the software check gets higher. It is remarkable that optimization has been introduced on the very same code that produced the unoptimized version, meaning that no further effort has been done to enhance error detection for the optimized version, so there is room for improvement. Even so, the data detection penalty is restrained and could be affordable for some applications.

In relation to error rates, optimized code has higher susceptibility to errors: 8.48% of injected faults produced errors while in the unoptimized version only 3.38% of injected faults resulted in error. Furthermore, data has demonstrated to be more prone to errors in the optimized version as 32.27% of total errors were data errors contrasting with the 21.93% of the unoptimized code. These two effects are related to a much higher use of registers in the case of the optimized version. Considering these results, it is interesting to go deeper on how registers are related to errors, extracting the injector information when the error occurs. Fig. 6.3 presents a comparison of the register sensitivity distribution in both code versions.

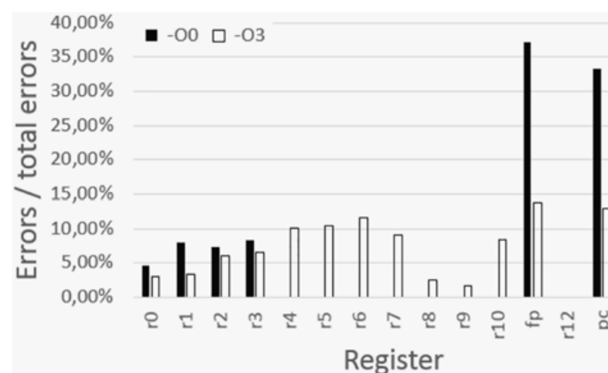

Fig. 6.3. Register sensitivity to errors.

Results in Fig. 6.3 demonstrate the more extensive usage of registers in the case of optimized (-O3) code, as all core registers except r12 produce errors. In the case of

unoptimized code, only four general purpose registers, r0 to r3, are used so faults injected on r4 to r10 have no impact on errors. Frame pointer (fp) and program counter (pc) are strongly related to the execution control-flow and have the highest error rates in both cases.

6.4. Conclusions

This work presents a Dual-Core Lockstep approach enhanced with redundant multithread support and control-flow error detection. Data error detection and recovering is based on the parallel execution of redundant threads on separate cores and a pure software technique that checks the correctness of the data and synchronizes the execution checks. Control-flow protection is accomplished by an external hardware IP that monitors the execution trace of the two cores in a non-intrusive way with small latency.

Experimental results demonstrate that control-flow errors are very likely, so that both data and control-flow checking are needed for effective error detection. The proposed approach achieves a high error detection rate (up to 99.63% error coverage) with low latency.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under projects ESP2015-68245-C4-1-P, ESP2015-68245-C4-3-P and by the Community of Madrid under grant IND2017/TIC-7776.

References

[J1] M. Peña-Fernandez *et al.*, “PTM-based hybrid error-detection architecture for ARM microprocessors”, *Microelectronics Reliability*, vol. 88-90, pp. 925–930, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.074.

[1] 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, pp. 293–306.

[3] M. Portela-García *et al.*, “On the use of embedded debug features for permanent and transient fault resilience in microprocessors”, *Microprocessors and Microsystems*, vol. 36, no. 5, pp. 334–343, 2012.

[27] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[73] S. K. Reinhardt and S. S. Mukherjee, "Transient fault detection via simultaneous multithreading", *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No. RS00201)*, pp. 25–36, 2000.

[74] T. Vijaykumar, I. Pomeranz, and K. Cheng, "Transient-fault recovery using simultaneous multithreading", *Proc. 29th Annual International Symposium on Computer Architecture*, pp. 38–87, 2002.

[76] M. Gomaa, C. Scarbrough, T. Vijaykumar, and I. Pomeranz, "Transient-fault recovery for chip multiprocessors", *Proc. 30th Annual International Symposium on Computer Architecture*, pp. 98–109, 2003.

[77] K.-H. Chen, G. von der Brüggen, and J.-J. Chen, "Reliability optimization on multi-core systems with multi-tasking and redundant multi-threading", *IEEE Transactions on Computers*, vol. 67, no. 4, pp. 484–497, Apr. 2018.

[82] F. Abate, L. Sterpone, and M. Violante, "A new mitigation approach for soft errors in embedded processors", *IEEE Transactions on Nuclear Science*, vol. 55, no. 4, pp. 2063–2069, Aug. 2008.

[83] X. Iturbe, B. Venu, E. Ozer, and S. Das, "A triple core lock-step (TCLS) ARM® cortex®-R5 processor for safety-critical and ultra-reliable applications", *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 246–249, 2016.

[87] J. Azambuja, S. Pagliarini, L. Rosa, and F. Kastensmidt, "Exploring the limitations of software-only techniques in see detection coverage", *Journal of Electronic Testing*, vol. 27, no. 4, pp. 541–550, 2011.

[118] ARM Inc, *CoreSight Components – Technical Reference Manual*. 2009.

[119] Xilinx Inc, *Zynq-7000 All Programmable SoC: Technical Reference Manual, UG585*. 2016.

[120] ARM Inc, *CoreSight Program Flow Trace Architecture Specification*. 2011.

[152] Xilinx Inc, *Soft error mitigation controller v4.1 Product guide, PG036*. 2014.

[154] X. Iturbe, B. Venu, and E. Ozer, "Soft error vulnerability assessment of the real-time safety-related ARM cortex-R5 CPU", *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 91–96, 2016.

[155] N. S. Bowen and D. K. Pradham, "Processor-and memory-based checkpoint and rollback recovery", *Computer*, vol. 26, no. 2, pp. 22–31, Feb. 1993.

[156] Á. B. de Oliveira *et al.*, "Lockstep dual-core ARM A9: Implementation and resilience analysis under heavy ion-induced soft errors", *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1783–1790, Aug. 2018.

[157] M. Violante, C. Meinhardt, R. Reis, and M. S. Reorda, "A low-cost solution for deploying processor cores in harsh environments", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 2617–2626, Jul. 2011.

[158] V. Bernon-Enjalbert, M. Blazy-Winning, R. Gubian, D. Lopez, J.-P. Meunier, and M. O'Donnell, "Safety integrated hardware solutions to support ASIL D applications", 2013.

[159] A. Shye, J. Blomstedt, T. Moseley, V. J. Reddi, and D. A. Connors, "PLR: A software approach to transient fault tolerance for multicore architectures", *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 2, pp. 135–148, Apr. 2009.

[160] H. Mushtaq, Z. Al-Ars, and K. Bertels, "Efficient software-based fault tolerance approach on multicore platforms", *Design, Automation & Test in Europe Conference & Exhibition*, pp. 921–926, 2013.

[161] G. S. Rodrigues, F. Rosa, Á. B. de Oliveira, F. L. Kastensmidt, L. Ost, and R. Reis, "Analyzing the impact of fault-tolerance methods in ARM processors under soft errors running Linux and parallelization APIs", *IEEE Transactions on Nuclear Science*, vol. 64, no. 8, pp. 2196–2203, Aug. 2017.

[162] ARM Inc, *Cortex-A9 MPCore Technical Reference Manual*. 2011.

[163] A. Serrano-Cases, F. Restrepo-Calle, S. Cuenca-Asensi, and A. Martínez-Álvarez, "Softerror mitigation for multi-core processors based on thread replication", *Proceedings of 20th IEEE Latin American Test Symposium*, Mar. 2019.

7. THE USE OF MICROPROCESSOR TRACE INFRASTRUCTURES FOR RADIATION-INDUCED FAULT DIAGNOSIS

Abstract

This work proposes a methodology to diagnose radiation-induced faults in a microprocessor using the hardware trace infrastructure. The diagnosis capabilities of this approach are demonstrated for an ARM microprocessor under neutron and proton irradiation campaigns. The experimental results demonstrate that the execution status in the precise moment that the error occurred can be reconstructed, so that error diagnosis can be achieved.

This chapter has been published as an article.

[J4] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "The use of microprocessor trace infrastructures for radiation-induced fault diagnosis", *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 126–134, Jan. 2020.

DOI: 10.1109/TNS.2019.2956204

URI: http://hdl.handle.net/10016/32699

8. ERROR DETECTION AND MITIGATION OF DATA-INTENSIVE MICROPROCESSOR APPLICATIONS USING SIMD AND TRACE MONITORING

Abstract

This article proposes a software error mitigation approach that uses the single instruction multiple data (SIMD) coprocessor to accelerate computation over redundant data. In addition, an external IP connected to the microprocessor's trace interface is used to detect errors that are difficult to cover with software-implemented techniques. The proposed approach has been implemented in an ARM microprocessor, and an irradiation campaign with neutrons has been carried out at Los Alamos National Laboratory. Experimental results demonstrate the high error coverage (more than 99.9%) of the proposed approach. The neutron cross section of errors that were not corrected nor detected was reduced by more than three orders of magnitude.

This chapter has been published as an article.

[J5] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "Error detection and mitigation of data-intensive microprocessor applications using SIMD and trace monitoring", *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1452–1460, Jul. 2020. DOI: 10.1109/TNS.2020.2992299

URI: http://hdl.handle.net/10016/32701

9. MICROPROCESSOR ERROR DIAGNOSIS BY TRACE MONITORING UNDER LASER TESTING

Abstract

This work explores the diagnosis capabilities of the enriched information provided by microprocessors trace subsystem combined with laser fault injection. Laser fault injection campaigns with delimited architectural regions have been accomplished on an ARM Cortex-A9 device. Experimental results demonstrate the capability of the presented technique to provide additional information of the various error mechanisms that can happen in a microprocessor. A comparison with radiation campaigns presented in previous work is also discussed, showing that laser fault injection results are in good agreement with neutron and proton radiation results.

This chapter has been published as an article.

[J6] M. Peña-Fernandez, A. Lindoso, L. Entrena, I. Lopes, and V. Pouget, "Microprocessor error diagnosis by trace monitoring under laser testing", *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1651–1659, Aug. 2021. DOI: 10.1109/TNS.2021.3067554

URI: http://hdl.handle.net/10016/35163

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

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.

This chapter has been published as an article.

[C1] M. Peña-Fernandez, A. Lindoso, and L. Entrena, "IP to detect and diagnose errors in COTS microprocessors through the Trace Interface", *presented at the 2nd European Workshop on On-Board Data Processing (OBDP2021)*, Jun. 2021. DOI: 10.5281/zenodo.5521538

10.1. 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 [27].

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 [90].

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 [46].

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 [92].

Single-event effects (SEEs) are a major concern in processors [145]. 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 [87].

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 [33].

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 10.2 summarizes the related work in the field. Section 10.3 describes the proposed technique. Section 10.4 illustrates some application cases. Finally, section 10.5 presents the conclusion of this work.

10.2. 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 [145]. 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 [65]. 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 [68].

Software control-flow hardening techniques rely on signatures or assertions [79] 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 [145]. 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 [87].

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 [67]. 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 [148]. 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 [156].

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 [46], 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) [53] 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 [104]. 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 [2] 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 [8]. 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 [1].

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 [118] 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.

10.3. 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) [118]. 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 [119], integrating a dual core ARM Cortex-A9 processor.

10.3.1. 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. 10.1.

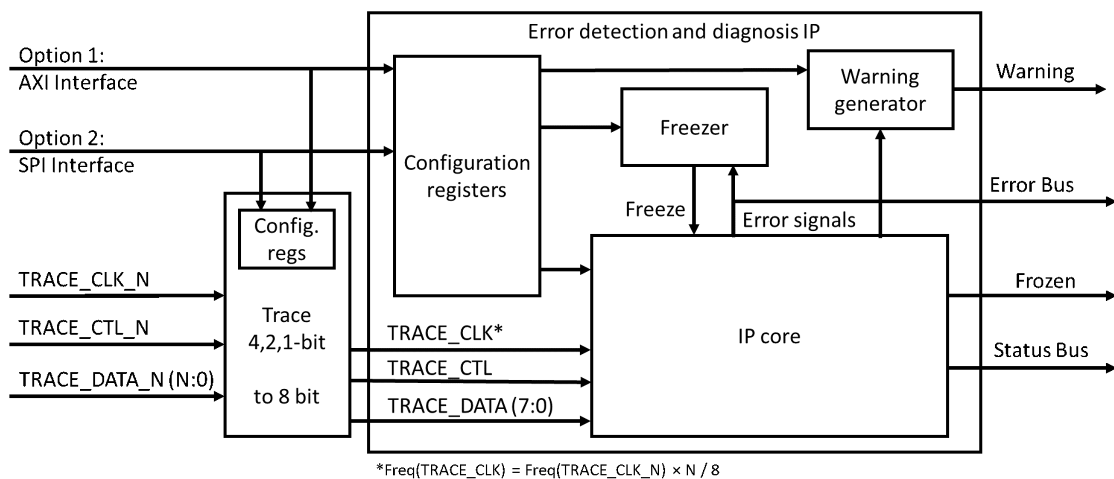

Fig. 10.1. Top level view of the IP and interfaces.

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.

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.

10.3.2. 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. 10.2. 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 10.3.3.
    - 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 10.3.3.

3. Checking resources examine the information received from the trace interface and, according to 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.

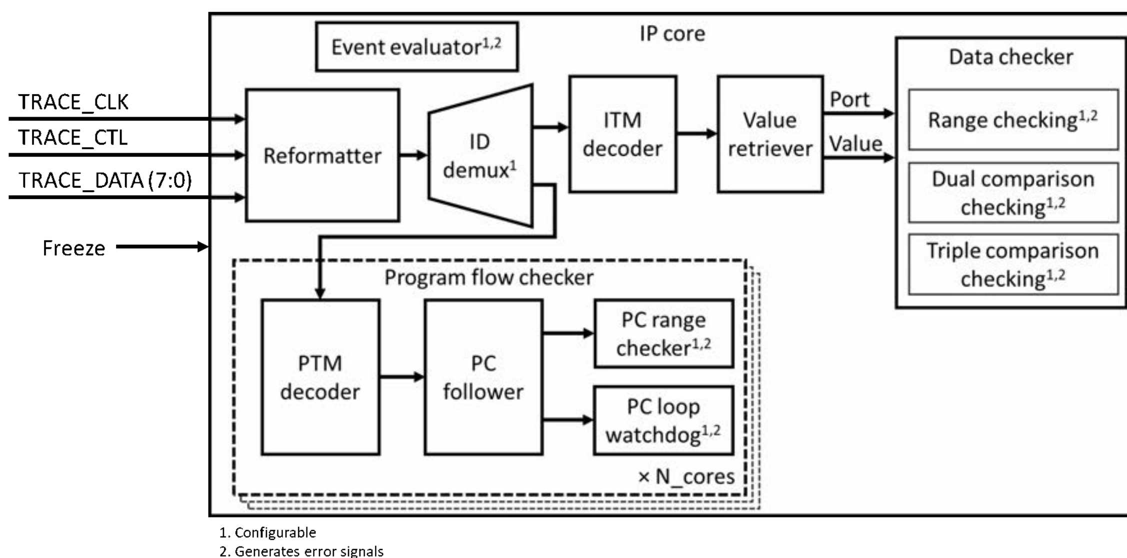

Fig. 10.2. Internal architecture view of the IP.

10.3.3. Checking resources

The error detection capabilities of the IP are defined by the integrated checking resources, represented in Fig. 10.2.

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.
- 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.

10.3.4. Error diagnosis

The information available at the trace interface is very rich, and the data rate for a typical application can exceed 1 Gbps. 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.

Table 10.1. 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
	@1333 Mbps	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	

10.4. 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 10.1. 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 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. 10.3 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. 10.3 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.

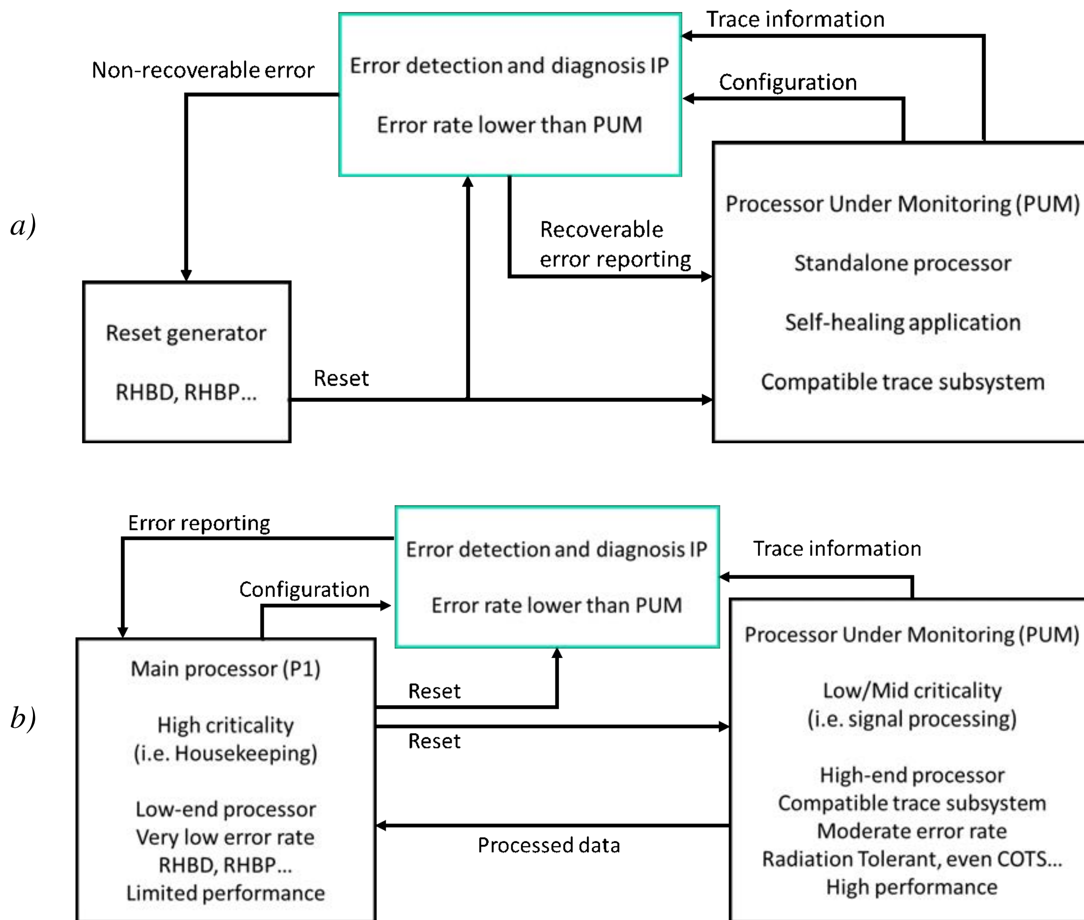

Fig. 10.3. IP integrated in a) binary and b) ternary architecture configuration.

[J1] and [J2] 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 [J3] and data redundancy acceleration using SIMD [J5], achieving up to 99.9% error coverage. Most recent works illustrate the error diagnosis capabilities of the IP under proton and neutron irradiation [J4] and also under laser fault injection [J6], demonstrating fine granularity on discriminating error types, and the suitability of the recorded information to perform effective error diagnosis.

10.5. 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%) [J5] and useful diagnosis information [J4, J6]. 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 University Carlos III of Madrid and Arquimea, in the framework of an Industrial Ph.D. program.

References

[J1] M. Peña-Fernandez *et al.*, "PTM-based hybrid error-detection architecture for ARM microprocessors", *Microelectronics Reliability*, vol. 88-90, pp. 925–930, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.074.

[J2] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martín-Holgado, "Online error detection through trace infrastructure in ARM microprocessors", *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1457–1464, Jul. 2019. DOI: 10.1109/TNS.2019.2921767.

[J3] M. Peña-Fernández *et al.*, "Dual-core lockstep enhanced with redundant multithread support and control-flow error detection", *Microelectronics Reliability*, vol. 100-101, no. 113447, Sep. 2019. DOI: 10.1016/j.microrel.2019.113447.

[J4] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "The use of microprocessor trace infrastructures for radiation-induced fault diagnosis", *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 126–134, Jan. 2020. DOI: 10.1109/TNS.2019.2956204.

[J5] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, “Error detection and mitigation of data-intensive microprocessor applications using SIMD and trace monitoring”, *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1452–1460, Jul. 2020. DOI: 10.1109/TNS.2020.2992299.

[J6] M. Peña-Fernandez, A. Lindoso, L. Entrena, I. Lopes, and V. Pouget, “Microprocessor error diagnosis by trace monitoring under laser testing”, *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1651–1659, Aug. 2021. DOI: 10.1109/TNS.2021.3067554.

[1] 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, pp. 293–306.

[2] M. Grosso, M. S. Reorda, M. Portela-Garcia, M. García-Valderas, C. López-Ongil, and L. Entrena, “An on-line fault detection technique based on embedded debug features”, *Proc. 16th IEEE International On-Line Testing Symposium*, pp. 167–172, 2010.

[8] A. Lindoso, L. Entrena, M. Garcia-Valderas, and L. Parra, “A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA”, *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 374–381, Jan. 2017.

[27] R. C. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies”, *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[33] H. Quinn, “Challenges in testing complex systems”, *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 766–786, Apr. 2014.

[46] K. A. LaBel, A. H. Johnston, J. L. Barth, R. A. Reed, and C. E. Barnes, “Emerging radiation hardness assurance (RHA) issues: A NASA approach for space flight programs”, *IEEE Transactions on Nuclear Science*, vol. 45, no. 6, pp. 2727–2736, Dec. 1998.

[53] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, “A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor”, *Proc. IEEE/ACM International Symposium on Microarchitecture*, pp. 29–40, Sep. 2003.

[65] P. Cheynet, B. Nicolescu, R. Velazco, M. Rebaudengo, M. Sonza Reorda, and M. Violante, “Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors”, *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2231–2236, 2000.

[67] A. Benso, S. Chiusano, P. Prinetto, and L. Tagliaferri, “A C/C++ source-to-source compiler for dependable applications”, *IEEE International Conference on Dependable Systems and Networks*, pp. 71–78, 2000.

[68] E. Chielle, J. R. Azambuja, R. S. Barth, F. Almeida, and F. L. Kastensmidt, "Evaluating selective redundancy in data-flow software-based techniques", *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2768–2775, Aug. 2013.

[79] M. Hiller, "Executable assertions for detecting data errors in embedded control systems", *Proc. International Conference on Dependable Systems and Networks*, pp. 24–33, 2000.

[87] J. Azambuja, S. Pagliarini, L. Rosa, and F. Kastensmidt, "Exploring the limitations of software-only techniques in see detection coverage", *Journal of Electronic Testing*, vol. 27, no. 4, pp. 541–550, 2011.

[90] R. Ginosar, "Survey of processors for space", *Proc. Int. Space System Engineering Conf. (DASIA)*, pp. 1–5, Aug. 2012.

[92] K. A. LaBel and M. J. Sampson, "NEPP roadmaps, COTS, and small missions", Jun. 2016.

[104] J. Mogollon, J. Nápoles, H. Guzman-Miranda, and M. Aguirre, "Real time SEU detection and diagnosis for safety or mission-critical ICs using hash library-based fault dictionaries", *Proc. 2011 12th European Conference on Radiation and Its Effects on Components and Systems*, vol. 3, pp. 705–710, Sep. 2011.

[118] ARM Inc, *CoreSight Components – Technical Reference Manual*. 2009.

[119] Xilinx Inc, *Zynq-7000 All Programmable SoC: Technical Reference Manual, UG585*. 2016.

[145] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, 2011.

[148] M. Pignol, "DMT and DT2: Two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers", *Proc. 12th IEEE International On-Line Testing Symposium*, 10–pp, Jul. 2006.

[156] Á. B. de Oliveira *et al.*, "Lockstep dual-core ARM A9: Implementation and resilience analysis under heavy ion-induced soft errors", *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1783–1790, Aug. 2018.

11. CONCLUSIONS AND FUTURE WORK

11.1. Conclusions

This Thesis proposes a new hardware monitoring approach for high-end, hard-core microprocessors using information available at the trace interface. The techniques developed in this Thesis improve the fault tolerance and the observability of microprocessor-based systems in the presence of radiation-induced faults that are applicable to space applications. The test vehicle for this work has been the Xilinx Zynq-7000 device family featuring a hard-core dual-core ARM Cortex-A9 microprocessor. Diverse techniques have been developed to leverage trace information concerning both execution control-flow and data values to detect and diagnose radiation-induced errors. Remarkably, most of the proposed techniques could be extensively applied to other ARM devices or even to other processor architectures.

The observation of the device is performed through the trace interface since it provides relevant information about execution in a non-intrusive manner. The selection of the trace interface as an observation point is highly convenient since it is a resource commonly available in modern microprocessors and is commonly left unused during system service life. The reuse of existing hardware for observation and error mitigation purposes is key when dealing with hard-core microprocessors, as their hardware cannot be modified. The trace interface of ARM microprocessors is based on the CoreSight technology, which introduces a modular subsystem for application trace and software debugging. In the particular case of the Zynq-7000 device, the main trace sources are the Program Trace Macrocell (PTM) and the Instrumentation Trace Macrocell (ITM) that generate trace information related to program execution and data values respectively. The information generated by the trace sources is handled by other CoreSight components and can be exported outside the processing system for user retrieval. Trace information is compressed and encoded before being output for an efficient use of the available bandwidth. Different ARM microprocessors may integrate different trace sources implementing different trace protocols, however they commonly include CoreSight components for program and data tracing producing similar information regardless of their type. In fact, CoreSight technology can be effectively used to tackle the observability challenges for complex microprocessor-based systems.

Given the high throughput of the trace interface, which can be above 1 Gbps, most part of the developed fault detection techniques have been implemented in dedicated hardware as a VHDL IP to enable online trace monitoring with minimum latency. Several hardware modules have been designed within the scope of this work and integrated into an IP capable to extract the trace information from the formatted data frames and encoded trace streams, based on the available CoreSight technical documentation. The IP includes additional

hardware modules for configuration and also checking resources capable to perform error detection and diagnosis using the retrieved information. Remarkably, the IP architecture has been oriented to obtain a modular and scalable design, capable of handling information coming from several trace sources. Its versatile design allows for simultaneous monitoring of control-flow and data or even multi-core system monitoring with a single IP. Scalability may enable the incorporation of further support for new trace sources in the future. The different IP capabilities are configurable and can be enabled or disabled through generic implementation parameters.

The control-flow error detection techniques developed in this work are based in the collection of the Program Counter (PC) value of the microprocessor during execution. Such information can be automatically exported by the PTM without interfering processor operation, thus introducing no performance penalty nor overhead in the application and no need to modify the code. The IP can detect errors in the PC value by checking whether it lies inside a specific memory region, that could be associated to the user application, or not. In the case that the processor is found to be executing code outside the expected region, a fault is signaled. By constantly checking the PC value it is possible to perform error detection with much lower latency than other common approaches, such as watchdog timers. Additionally, another error detection technique was developed to detect whether the processor is not executing the application main loop in the expected timely manner, by checking the elapsed time between consecutive iterations. Such techniques are highly effective and present low complexity, and have proved to detect a very high portion of control-flow errors in the different fault injection and irradiation campaigns performed in this work.

Trace-based control-flow error detection techniques developed in this Thesis have been succesfully combined with a software-based dual-core lockstep technique in collaboration with Alicante University to form a hybrid technique. The hybrid approach leverages the dual-core processing system (PS) present in the Zynq-7000 device to detect and correct data errors by executing the same program in both available processors and comparing between results. In the case of mismatch, a re-execution is triggered to obtain a correct result. It is remarkable the concept of reusing existing hardware, such as a spare processor in a dual-core system, to introduce fault-tolerance techniques. The control-flow errors are detected in this approach through the trace information available from both processors, which is simultaneously gathered by the IP to check the PC values of each one. The contribution of trace monitoring in this approach is crucial, since the detection of control-flow errors is a complex challenge for the software-implemented dual-core lockstep system, but it is much more affordable for the trace checker IP. This approach has demonstrated to achieve high error coverage both in fault injection and under proton irradiation testing.

Data error detection techniques have also been developed in this work by leveraging the capability of the ITM to export through the trace the value of any data variable in the software during execution. For the data value to be exported, the value shall be

explicitly written by the application in a set of specific-purpose registers. Thus, some modification is needed in the application code that could lead to an associated overhead in performance and code size. Nevertheless, simple rules can be developed to introduce the new instructions corresponding to those points in the code before the computation results are output, thus limiting the overhead. The IP can detect errors on data values by checking whether they are within a predefined expected range or by performing consistency checks on replicated data. In the case of data replication, the code must be modified to support the replication of redundant data variables, introducing performance penalty, but the IP enables the possibility to perform the consistency checks on hardware, thus reducing the total performance penalty when compared with a pure-software replication approach. Such low complexity checks have been proved to detect a high portion of data errors in the performed fault injection and irradiation campaigns with low latency.

Data error detection techniques developed in this Thesis have been successfully combined with a SIMD-based data error mitigation technique with replicated data, conforming a hybrid technique. This hybrid approach leverages the capability of NEON SIMD engine present in the processing system (PS) of the Zynq-7000 device to perform simultaneous computations on multiple data with a single instruction. This acceleration scheme is used to detect and correct data errors by performing replicated computations on triplicated data with very low performance penalty. It is remarkable again the concept of reusing the existing resources available on the hardware to introduce fault-tolerance techniques. The trace information is used in this approach to detect control-flow errors in the execution and also to perform the consistency checks on replicated data, removing the burden of performing such complex tasks at the software. This approach has demonstrated to achieve very high error detection and correction capabilities in fault injection and under neutron irradiation testing.

The different fault detection techniques presented in this work have been progressively developed and introduced in the IP during the evolution of the performed research. Different tests have been carried out in this work allowing to validate the developed techniques and obtain intermediate results to make decisions about research directions and evaluation of the capabilities of the proposed approach. Software fault injection has been used throughout the entire work to continuously support the development of the different techniques at desktop level and to perform preliminary evaluation before validation campaigns. Irradiation campaigns have been performed at external renowned facilities at key points during development to validate the developed techniques under radiation-induced faults and obtain results susceptible to publication. A total of three low-energy proton irradiation campaigns have been performed in Centro Nacional de Aceleradores (CNA) in Spain, one campaign with neutron irradiation in Los Alamos Neutron Science Center (LANSCE) in the United States and two campaigns with laser fault injection in University of Montpellier in France. All of them have enabled to prove the validity of our developments and to find new opportunities to improve the work.

One of the most relevant findings during the development of this Thesis was the use of the trace information not only to perform radiation-induced error detection but also error diagnosis. Although it was not initially planned, error diagnosis was proposed based on the results obtained in the first test campaigns, which suggested that trace information could be used to contextualize the error instant and to discriminate between different error types. Error diagnosis has gained high relevance in this work and specific techniques have been developed to obtain useful diagnosis information from the trace. Dedicated fault injection and irradiation campaigns have been performed to evaluate the potential diagnosis capabilities of trace information demonstrating that the trace interface provides enough observability to obtain detailed information about the observed errors and perform effective error diagnosis. The obtained diagnosis information can be used to analyze the causes of radiation-induced errors and to identify weak points in microprocessor-based applications, allowing to design optimal error mitigation solutions. The diagnosis techniques presented in this work have demonstrated an unprecedented accuracy in the performed fault injection and irradiation campaigns, allowing to precisely locate faults and to find error patterns. Laser fault injection campaigns have shown that the proposed error diagnosis techniques can enable the identification of zones more prone to errors than others in a given device. In addition, the proposed diagnosis approach is non-intrusive, presenting high potential for its use in hard-core microprocessors performing critical applications.

It is remarkable that the techniques proposed in this work have obtained very high error detection and detailed error diagnosis capabilities while dealing with a complex information resource such as the ARM CoreSight trace interface. In fact, the fixed implementation of the CoreSight subsystem in the device actually limits the available information and conditions the possible error detection and diagnosis approaches. While other works devoted to microprocessor fault detection through trace monitoring for soft-core microprocessors may have access to a richer, or even custom trace interface, this work is the first one intending to detect radiation-induced faults on a complex hard-core microprocessor with a standard, fixed trace interface imposing observability restrictions. This is a pioneer work intended to perform detection and diagnosis of radiation-induced errors in hard-core microprocessors through the trace interface, dealing with both control-flow erros and data errors. All proposed techniques have been tested under fault injection and irradiation test campaigns to evaluate their effectiveness and limitations. Obtained results demonstrate the feasibility of the approach, that could potentially contribute to risk management applied to applications within the New Space paradigm.

In the academic and research field, this work has received excellent acceptance at renowned specialized forums in the area of radiation effects in electronic circuits and devices. Four contributions were communicated and accepted in four consecutive editions of the most relevant European conference in the field, namely RADECS (Radiation and its Effects on Components and Systems), with one poster in 2018 and three oral presentations performed by the author in 2019, 2020 and 2021. One contribution was also accepted as a poster at the most important American conference in this field, namely NSREC (Nuclear

and Space Radiation Effects Conference) in 2020. RADECS and NSREC are the most important conferences of the Nuclear and Plasma Sciences Society of the IEEE. Another relevant European conference in the reliability topic is ESREF (European Symposium on Reliability of Electron Devices, Failure Physics and Analysis), where two communications were accepted as oral presentations performed by the author in 2018 and 2019, respectively. Seven publications have been attained in prestigious journals, including IEEE Transactions on Nuclear Science (JCR Q2) with 5 publications and Microelectronics Reliability (JCR Q3) with 2 publications.

This Thesis has been carried out in the framework of an Industrial Ph.D. program, supported by the Community of Madrid under Grant IND2017/TIC-7776 awarded in 2017. By this Industrial Ph.D. the aerospace company Arquimea and the University Carlos III of Madrid (UC3M) have collaborated in the development of this work and promoted the transfer of technology from the academia to the industry. Particularly, UC3M has led the research activities while Arquimea has pursued their implementation as a marketable product; and the author has carried out the work inside the company environment. For all involved parties, this Thesis has provided visibility in international forums and has increased mutual collaboration, augmenting the possibilities of future projects in partnership.

This work has been carried out gradually by developing and testing different error detection and diagnosis techniques. However, none of them have been tested separately from the others, but they have been progressively integrated in a common design as a single IP. As a result, the obtained IP has become modular, scalable and configurable to tune its functionality depending on the test. The developed hardware has the potential to be adapted to different implementations and scenarios, as illustrated by the different chapters in this work, so it has potential to be used by third parties for research or industrial purposes. Moreover, the developed techniques can be easily integrated with other microprocessor fault-tolerance techniques to form hybrid techniques as demonstrated in this work. In addition, the IP requires minimal effort and software support to be used, since it only needs initial configuration to start operating independently of the processor, checking its behavior by the trace interface. Given its industrial orientation, the results of this Thesis have actually been communicated to the On-Board Data Processing workshop OBDP2021 organized by ESA in mid-2021. An oral presentation was made introducing the developed IP as a product available to customers to be included in the life cycle of space designs to increase observability of microprocessor-based systems during testing and to detect errors during operation in space.

A very relevant result of this Industrial Ph.D is that it has notably promoted additional R&D activities in the company, focusing on new technologies applied to error mitigation for COTS microprocessors under radiation. As a result, the error detection and diagnosis IP concept has also been presented to European Space Agency through a call of ideas for technology transfer related to COTS, called Open Space Innovation Platform (OSIP). The idea was positively informed and was selected for becoming a collaboration project

between Arquimea and University Carlos III of Madrid (UC3M) funded by ESA, which has already started in late-2021. The achievement of this project demonstrates the industrial interest on the technology developed in this Thesis. The project intends to give continuity to the developed technology to become an industrially available product. The new project, which is out of the scope of this Thesis, takes this Thesis as a starting point and proposes to focus on the development and evaluation of an error detection and diagnosis tool according to the current needs of New Space sector. It is remarkable that the technology transfer, which was an important objective given the Industrial orientation of this work, has been successfully achieved by the attainment of this new project, sponsored by an important actor in the space sector such as the European Space Agency. The achievement of this contract reveals that a research result, backed by several publications, has been successfully transferred and exploited to the industry with benefit for the company, which demonstrates the accomplishment of the ultimate objective of the Industrial Ph.D. program.

11.2. Future work

This work has set the foundations for further activities by demonstrating the feasibility and potential applications and capabilities of error detection and diagnosis techniques based on trace information for complex, hard-core microprocessors. From the experience acquired from this work, some future activities have been identified and are proposed hereunder. Many of them are currently being addressed at the company as part of the new OSIP project funded by ESA:

- **Expand IP capabilities to support more ARM microprocessors and devices.** Currently the IP is compatible with the Program Trace Macrocell (PTM) and the Instrumentation Trace Macrocell (ITM) trace sources, and has been tested with Xilinx Zynq-7000 device family. However, the IP could potentially be compatible with any other CoreSight trace source such as the Embedded Trace Macrocell (ETM) or the System Trace Macrocell (STM), and support a higher number of devices. The modular and scalable architecture of the IP allows to design a configurable module that could be adapted to different implementation scenarios. This action is currently part of the OSIP project with European Space Agency, with the objective to identify the ARM technologies and devices of interest in the space industry and to make the IP compatible with them.

- **Validation under heavy ions irradiation.** Despite the techniques developed within this work have been validated under proton and neutron irradiation, currently the gold standard for Single-Event Effects in space community is heavy ion testing. It has not been possible to perform such testing within the scope of this work due to budget and calendar limitations, combined with the COVID-19 pandemic outbreak, that severely limited mobility and access to such facilities. This action is currently

part of the OSIP project with European Space Agency, planning to perform a final technology validation under heavy ion testing.

- **Evaluation with high complexity software applications.** The software applications used as benchmarks within this Thesis have been selected to have reduced complexity to increase the controlability of the system during fault injection and irradiation campaigns. This approach has helped to have a good control and understanding of the experimental setup, allowing to obtain as much information as possible to improve the design. However, the applications commonly run by microprocessors in space are far more complex such as data processing, data compression or image processing. Increasing the complexity of the used benchmarks would lead to a different scenario and reveal new challenges enabling future developments. This action is currently part of the OSIP project with European Space Agency, planning to perform the final validation campaigns with at least one complex benchmark representative of a space application.

- **Develop a commercial product.** The IP resulting from this work could be considered as a research tool to evaluate the capabilities of the developed techniques and the potential of trace information to be used for error detection and diagnosis. However, the current design is not conceived for standard users that need guidance on how to build applications around it. By performing an extensive design review and documentation, a wider range of potential users could benefit of the innovative capabilities of this technology. In fact, this is a crucial step to reach an industrial phase of the IP. These actions are currently part of the new development project with European Space Agency, willing to obtain a marketable IP design and commercial documentation for potential users and customers.

- **Develop, implement and evaluate additional checking techniques.** The error detection techniques developed within this Thesis have demonstrated to cover a high amount of errors in validation campaigns, while maintaining low complexity and requiring few resources to be implemented. Thanks to the proposed techniques described in this Thesis, additional research can be carried out to develop more complex techniques, especially by combining the information from different trace sources.

BIBLIOGRAPHY

[J1] M. Peña-Fernandez *et al.*, "PTM-based hybrid error-detection architecture for ARM microprocessors", *Microelectronics Reliability*, vol. 88-90, pp. 925–930, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.074.

[J2] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martín-Holgado, "Online error detection through trace infrastructure in ARM microprocessors", *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1457–1464, Jul. 2019. DOI: 10.1109/TNS.2019.2921767.

[J3] M. Peña-Fernández *et al.*, "Dual-core lockstep enhanced with redundant multithread support and control-flow error detection", *Microelectronics Reliability*, vol. 100-101, no. 113447, Sep. 2019. DOI: 10.1016/j.microrel.2019.113447.

[J4] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "The use of microprocessor trace infrastructures for radiation-induced fault diagnosis", *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 126–134, Jan. 2020. DOI: 10.1109/TNS.2019.2956204.

[J5] M. Peña-Fernandez, A. Lindoso, L. Entrena, and M. Garcia-Valderas, "Error detection and mitigation of data-intensive microprocessor applications using SIMD and trace monitoring", *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1452–1460, Jul. 2020. DOI: 10.1109/TNS.2020.2992299.

[J6] M. Peña-Fernandez, A. Lindoso, L. Entrena, I. Lopes, and V. Pouget, "Microprocessor error diagnosis by trace monitoring under laser testing", *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1651–1659, Aug. 2021. DOI: 10.1109/TNS.2021.3067554.

[J7] M. Peña-Fernández *et al.*, "Hybrid lockstep technique for soft error mitigation", *IEEE Transactions on Nuclear Science*, 2022. DOI: 10.1109/TNS.2022.3149867.

[C1] M. Peña-Fernandez, A. Lindoso, and L. Entrena, "IP to detect and diagnose errors in COTS microprocessors through the Trace Interface", *presented at the 2nd European Workshop on On-Board Data Processing (OBDP2021)*, Jun. 2021. DOI: 10.5281/zenodo.5521538.

[1] 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, pp. 293–306.

[2] M. Grosso, M. S. Reorda, M. Portela-Garcia, M. García-Valderas, C. López-Ongil, and L. Entrena, "An on-line fault detection technique based on embedded debug features", *Proc. 16th IEEE International On-Line Testing Symposium*, pp. 167–172, 2010.

[3] M. Portela-García *et al.*, "On the use of embedded debug features for permanent and transient fault resilience in microprocessors", *Microprocessors and Microsystems*, vol. 36, no. 5, pp. 334–343, 2012.

[4] L. Parra *et al.*, "Efficient mitigation of data and control flow errors in microprocessors", *IEEE Transactions on Nuclear Science*, vol. 61, no. 4, pp. 1590–1596, 2014.

[5] L. Parra *et al.*, "A new hybrid nonintrusive error-detection technique using dual control-flow monitoring", *IEEE Transactions on Nuclear Science*, vol. 61, no. 6, pp. 3236–3243, 2014.

[6] L. Entrena *et al.*, "Flexible approaches to fault-tolerant microprocessors for space applications", *Proc. Data Systems in Aerospace (DASIA), ESA Special Publication SP-732*, May 2015.

[7] B. Du *et al.*, "Online test of control flow errors: A new debug interface-based approach", *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1846–1855, Jun. 2016.

[8] A. Lindoso, L. Entrena, M. Garcia-Valderas, and L. Parra, "A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA", *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 374–381, Jan. 2017.

[9] B. Johnson, "An introduction to the design and analysis of fault-tolerant systems", pp. 1–87, 1996.

[10] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[11] G. Buja and R. Menis, "Dependability and functional safety: Applications in industrial electronics systems", *IEEE Industrial Electronics Magazine*, vol. 6, no. 3, pp. 4–12, 2012.

[12] A. Avizienis, J.-C. Laprie, B. Randell, *et al.*, *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[13] W. Kuo, W.-T. K. Chien, and T. Kim, *Reliability, yield, and stress burn-in: a unified approach for microelectronics systems manufacturing & software development*. Springer Science & Business Media, 1998.

[14] J. Lienig and H. Bruemmer, *Fundamentals of electronic systems design*. Springer, 2017.

[15] R. C. Lacoe, "Improving integrated circuit performance through the application of hardness-by-design methodology", *IEEE Transactions on Nuclear Science*, vol. 55, no. 4, pp. 1903–1925, Sep. 2008.

[16] J. D. Cressler, "Big picture and some history of the field", in *Extreme Environment Electronics*, J. D. Cressler and H. A. Mantooth, Eds., Boca Raton: CRC Press, 2013, pp. 3–9.

[17] E. G. Stassinopoulos and J. P. Raymond, “The space radiation environment for electronics”, *Proceedings of the IEEE*, vol. 76, no. 11, pp. 1423–1442, Nov. 1988.

[18] M. Xapsos, “A brief history of space climatology: From the big bang to the present”, *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 17–37, Jan. 2019.

[19] P. K. Grieder, *Cosmic rays at Earth*. Elsevier, 2001.

[20] L. Dilillo, F. Wrobel, J. Galliere, and F. Saigne, “Neutron detection through an SRAM-based test bench”, *Proc. IEEE Int. Workshop Advanced Sensor and Interfaces (IWASI)*, pp. 64–69, 2009.

[21] D. Heynderickx, B. Quaghebeur, E. Speelman, and E. Daly, “ESA’s space environment information system (SPENVIS)-a WWW interface to models of the space environment and its effects”, in *38th Aerospace Sciences Meeting and Exhibit*, 2000, p. 371.

[22] T. R. Oldham and F. McLean, “Total ionizing dose effects in MOS oxides and devices”, *IEEE transactions on nuclear science*, vol. 50, no. 3, pp. 483–499, Jun. 2003.

[23] D. M. Fleetwood, “Total ionizing dose effects in MOS and low-dose-rate-sensitive linear-bipolar devices”, *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1706–1730, Jun. 2013.

[24] J. Srour and J. Palko, “Displacement damage effects in irradiated semiconductor devices”, *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1740–1766, Jun. 2013.

[25] P. E. Dodd and L. W. Massengill, “Basic mechanisms and modeling of single-event upset in digital microelectronics”, *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 583–602, Jun. 2003.

[26] F. W. Sexton, “Destructive single-event effects in semiconductor devices and ICs”, *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 603–621, Jun. 2003.

[27] R. C. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies”, *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[28] P. E. Dodd, M. R. Shaneyfelt, J. R. Schwank, and J. A. Felix, “Current and future challenges in radiation effects on CMOS electronics”, *IEEE Transactions on Nuclear Science*, vol. 57, no. 4, pp. 1747–1763, 2010.

[29] R. Baumann, “Soft errors in advanced computer systems”, *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, Jun. 2005.

[30] M. L. Alles, “Radiation hardening by process”, in *Extreme Environment Electronics*, J. D. Cressler and H. A. Mantooth, Eds., Boca Raton: CRC Press, 2013, pp. 287–297.

[31] European Space Agency, *Space product assurance. Techniques for radiation effects mitigation in ASICs and FPGAs handbook*. ECSS, 2016.

[32] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron CMOS technology", *IEEE Transactions on nuclear science*, vol. 43, no. 6, pp. 2874–2878, Dec. 1996.

[33] H. Quinn, "Challenges in testing complex systems", *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 766–786, Apr. 2014.

[34] J. R. Schwank, M. R. Shaneyfelt, and P. E. Dodd, "Radiation hardness assurance testing of microelectronic devices and integrated circuits: Radiation environments, physical mechanisms, and foundations for hardness assurance", *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 2074–2100, Jun. 2013.

[35] Department of Defense, *Test Method Standard Microcircuits MIL-STD-883*. 2013.

[36] European Space Agency, *Total dose steady-state irradiation test method*. 2010.

[37] European Space Agency, *Single event effects test method and guidelines*. 2014.

[38] J. E. D. E. C. (JEDEC), *Test Procedure for the Management of Single-Event Effects in Semiconductor Devices from Heavy Ion Irradiation, JESD57*. 2017.

[39] J. E. D. E. C. (JEDEC), *Test Standard For The Measurement of Proton Radiation Single Event Effects in Electronic Devices, JESD234*. 2013.

[40] H. Quinn, T. Fairbanks, J. L. Tripp, G. Duran, and B. Lopez, "Single-event effects in low-cost, low-power microprocessors", *2014 IEEE Radiation Effects Data Workshop (REDW)*, pp. 1–9, 2014.

[41] H. M. Quinn, D. A. Black, W. H. Robinson, and S. P. Buchner, "Fault simulation and emulation tools to augment radiation-hardness assurance testing", *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 2119–2142, Jun. 2013.

[42] S. P. Buchner, F. Miller, V. Pouget, and D. P. McMorrow, "Pulsed-laser testing for single-event effects investigations", *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1852–1875, Jun. 2013.

[43] K. Sahu, "EEE-INST-002: Instructions for EEE parts selection, screening, qualification, and derating", *NASA/TP—2003–212242*, 2008.

[44] R. Baumann, "Radiation hardness assurance in the 'Wild West' of commercial space", in *IEEE NSREC Short Course*, Dec. 2020, pp. IV/1–IV/68.

[45] NASA Engineering & Safety Center, "COTS components in spacecraft systems: Understanding the risk", *Reference NASA/TM-2014-218261*,

[46] K. A. LaBel, A. H. Johnston, J. L. Barth, R. A. Reed, and C. E. Barnes, "Emerging radiation hardness assurance (RHA) issues: A NASA approach for space flight programs", *IEEE Transactions on Nuclear Science*, vol. 45, no. 6, pp. 2727–2736, Dec. 1998.

[47] J. Plante and M. Sampson, "Cost impacts of upgrading electronic parts for use in NASA spaceflight systems", 2003.

[48] P. Winokur, G. Lum, M. Shaneyfelt, F. Sexton, G. Hash, and L. Scott, "Use of COTS microelectronics in radiation environments", *IEEE Transactions on Nuclear Science*, vol. 46, no. 6, pp. 1494–1503, Dec. 1999.

[49] H. Quinn, Z. Baker, T. Fairbanks, J. L. Tripp, and G. Duran, "Software resilience and the effectiveness of software mitigation in microcontrollers", *IEEE Transactions on Nuclear Science*, vol. 62, no. 6, pp. 2532–2538, Dec. 2015.

[50] F. Irom, "Guideline for ground radiation testing of microprocessors in the space radiation environment", Jet Propulsion Laboratory, National Aeronautics and Space Administration, Pasadena, CA, Tech. Rep., 2008.

[51] S. M. Guertin, "Guideline for single-event effect (SEE) testing of system on a chip (SOC) devices", Jet Propulsion Laboratory, National Aeronautics and Space Administration, Pasadena, CA, Tech. Rep., 2018.

[52] H. Quinn *et al.*, "Using benchmarks for radiation testing of microprocessors and FPGAs", *IEEE Transactions on Nuclear Science*, vol. 62, no. 6, pp. 2547–2554, Dec. 2015.

[53] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor", *Proc. IEEE/ACM International Symposium on Microarchitecture*, pp. 29–40, Sep. 2003.

[54] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, "Multi-bit upsets vulnerability analysis of modern microprocessors", *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 119–130, 2019.

[55] A. Serrano-Cases, L. M. Reyneri, Y. Morilla, S. Cuenca-Asensi, and A. Martínez-Álvarez, "Empirical mathematical model of microprocessor sensitivity and early prediction to proton and neutron radiation-induced soft errors", *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1511–1520, Jul. 2020.

[56] R. Velazco, S. Rezgui, and R. Ecoffet, "Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulating Upsets) injection", *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2405–2411, Dec. 2000.

[57] M. Rebaudengo, M. S. Reorda, and M. Violante, "Software-level soft-error mitigation techniques", in *Soft Errors in Modern Electronic Systems*, Springer, 2011.

[58] V. Nair, H. Kim, N. Krishnamurthy, and J. Abraham, “Design and evaluation of automated high-level checks for signal processing applications”, *Proc. SPIE advanced algorithms and architectures for signal processing conference*, Aug. 1996.

[59] Z. Alkhalifa, V. S. Nair, N. Krishnamurthy, and J. A. Abraham, “Design and evaluation of system-level checks for on-line control flow error detection”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 627–641, Jun. 1999.

[60] N. Oh, P. P. Shirvani, and E. J. McCluskey, “Control-flow checking by software signatures”, *IEEE transactions on Reliability*, vol. 51, no. 1, pp. 111–122, Mar. 2002.

[61] O. Goloubeva, M. Rebaudengo, M. Sonza Reorda, and M. Violante, “Soft-error detection using control flow assertions”, *Proc. 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 581–588, 2003.

[62] R. Vemu and J. A. Abraham, “CEDA: Control-flow error detection through assertions”, *Proc. 12th IEEE International On-Line Testing Symposium (IOLTS)*, pp. 151–158, 2006.

[63] J. R. Azambuja *et al.*, “A fault tolerant approach to detect transient faults in microprocessors based on a non-intrusive reconfigurable hardware”, *IEEE Transactions on Nuclear Science*, vol. 59, no. 4, pp. 1117–1124, Aug. 2012.

[64] M. Rebaudengo, M. S. Reorda, M. Torchiano, and M. Violante, “Soft-error detection through software fault-tolerance techniques”, in *Proceedings 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (EFT’99)*, Nov. 1999, pp. 210–218.

[65] P. Cheynet, B. Nicolescu, R. Velazco, M. Rebaudengo, M. Sonza Reorda, and M. Violante, “Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors”, *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2231–2236, 2000.

[66] B. Nicolescu and R. Velazco, “Detecting soft errors by a purely software approach: Method, tools and experimental results”, *Design, Automation and Test in Europe Conference*, pp. 57–62, 2003.

[67] A. Benso, S. Chiusano, P. Prinetto, and L. Tagliaferri, “A C/C++ source-to-source compiler for dependable applications”, *IEEE International Conference on Dependable Systems and Networks*, pp. 71–78, 2000.

[68] E. Chielle, J. R. Azambuja, R. S. Barth, F. Almeida, and F. L. Kastensmidt, “Evaluating selective redundancy in data-flow software-based techniques”, *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2768–2775, Aug. 2013.

[69] N. Oh, P. P. Shirvani, and E. J. McCluskey, "Error detection by duplicated instructions in super-scalar processors", *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 63–75, Mar. 2002.

[70] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, and D. I. August, "SWIFT: Software implemented fault tolerance", *Proc. 3rd International Symposium on Code Generation and Optimization (CGO)*, pp. 243–254, Mar. 2005.

[71] G. A. Reis, J. Chang, and D. I. August, "Automatic instruction-level software-only recovery", *IEEE micro*, vol. 27, no. 1, pp. 36–47, Jan. 2007.

[72] N. Oh and E. J. McCluskey, "Error detection by selective procedure call duplication for low energy consumption", *IEEE Transactions on Reliability*, vol. 51, no. 4, pp. 392–402, Dec. 2002.

[73] S. K. Reinhardt and S. S. Mukherjee, "Transient fault detection via simultaneous multithreading", *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No. RS00201)*, pp. 25–36, 2000.

[74] T. Vijaykumar, I. Pomeranz, and K. Cheng, "Transient-fault recovery using simultaneous multithreading", *Proc. 29th Annual International Symposium on Computer Architecture*, pp. 38–87, 2002.

[75] S. Mukherjee, *Architecture design for soft errors*. Elsevier, 2008.

[76] M. Gomaa, C. Scarbrough, T. Vijaykumar, and I. Pomeranz, "Transient-fault recovery for chip multiprocessors", *Proc. 30th Annual International Symposium on Computer Architecture*, pp. 98–109, 2003.

[77] K.-H. Chen, G. von der Brüggen, and J.-J. Chen, "Reliability optimization on multi-core systems with multi-tasking and redundant multi-threading", *IEEE Transactions on Computers*, vol. 67, no. 4, pp. 484–497, Apr. 2018.

[78] N. Oh, S. Mitra, and E. J. McCluskey, "ED4I: Error detection by diverse data and duplicated instructions", *IEEE Transactions on Computers*, vol. 51, no. 2, pp. 180–199, Feb. 2002.

[79] M. Hiller, "Executable assertions for detecting data errors in embedded control systems", *Proc. International Conference on Dependable Systems and Networks*, pp. 24–33, 2000.

[80] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA design robustness with partial TMR", in *2006 IEEE International Reliability Physics Symposium Proceedings*, Mar. 2006, pp. 226–232.

[81] A. J. Sánchez-Clemente, L. Entrena, and M. García-Valderas, "Partial TMR in FPGAs using approximate logic circuits", *IEEE Transactions on Nuclear Science*, vol. 63, no. 4, pp. 2233–2240, Jul. 2016.

[82] F. Abate, L. Sterpone, and M. Violante, "A new mitigation approach for soft errors in embedded processors", *IEEE Transactions on Nuclear Science*, vol. 55, no. 4, pp. 2063–2069, Aug. 2008.

[83] X. Iturbe, B. Venu, E. Ozer, and S. Das, "A triple core lock-step (TCLS) ARM® cortex®-R5 processor for safety-critical and ultra-reliable applications", *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 246–249, 2016.

[84] P. Bernardi, L. B. Poehls, M. Grosso, and M. S. Reorda, "A hybrid approach for detection and correction of transient faults in SoCs", *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 439–445, Aug. 2010.

[85] A. Benso, S. Di Carlo, G. Di Natale, and P. Prinetto, "A watchdog processor to detect data and control flow errors", in *9th IEEE On-Line Testing Symposium*, 2003, pp. 144–148.

[86] S. Bergaoui and R. Leveugle, "IDSM: An improved control flow checking approach with disjoint signature monitoring", *roc. 24th Conference on Design of Circuits and Integrated Systems (DCIS)*, 2009.

[87] J. Azambuja, S. Pagliarini, L. Rosa, and F. Kastensmidt, "Exploring the limitations of software-only techniques in see detection coverage", *Journal of Electronic Testing*, vol. 27, no. 4, pp. 541–550, 2011.

[88] J. R. Azambuja, M. Altieri, J. Becker, and F. L. Kastensmidt, "HETA: Hybrid error-detection technique using assertions", *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2805–2812, 2013.

[89] I. V. McLoughlin, V. Gupta, G. Sandhu, S. Lim, and T. Bretschneider, "Fault tolerance through redundant COTS components for satellite processing applications", in *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, vol. 1, Dec. 2003, pp. 296–299.

[90] R. Ginosar, "Survey of processors for space", *Proc. Int. Space System Engineering Conf. (DASIA)*, pp. 1–5, Aug. 2012.

[91] T. M. Lovelly and A. D. George, "Comparative analysis of present and future space-grade processors with device metrics", *Journal of aerospace information systems*, vol. 14, no. 3, pp. 184–197, 2017.

[92] K. A. LaBel and M. J. Sampson, "NEPP roadmaps, COTS, and small missions", Jun. 2016.

[93] R. Trautner, "ESA's roadmap for next generation payload data processors", in *Proc. Data Syst. Aerosp. Conf.(DASIA)*, 2011, pp. 159–161.

[94] P. P. Shirvani, N. Oh, E. J. Mccluskey, D. Wood, M. N. Lovellette, and K. Wood, "Software-implemented hardware fault tolerance experiments: COTS in space", in *International Conference on Dependable Systems and Networks (FTCS-30 and DCCA-8), New York (NY)*, 2000.

[95] J. R. Marshall, "Embedding COTS processors into fault tolerant space applications", in *10th Computing in Aerospace Conference*, 1995, p. 1032.

[96] D. Gizopoulos *et al.*, “Architectures for online error detection and recovery in multicore processors”, in *2011 Design, Automation & Test in Europe*, IEEE, 2011, pp. 1–6.

[97] D. S. Khudia, G. Wright, and S. Mahlke, “Efficient soft error protection for commodity embedded microprocessors using profile information”, *ACM SIGPLAN Notices*, vol. 47, no. 5, pp. 99–108, 2012.

[98] T. A. Alves, S. Kundu, L. A. Marzulo, and F. M. França, “Online error detection and recovery in dataflow execution”, in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, IEEE, 2014, pp. 9–104.

[99] R. Vemu, S. Gurumurthy, and J. A. Abraham, “ACCE: Automatic correction of control-flow errors”, in *2007 IEEE International Test Conference*, IEEE, 2007, pp. 1–10.

[100] N. Khoshavi, H. R. Zarandi, and M. Maghsoudloo, “Two control-flow error recovery methods for multithreaded programs running on multi-core processors”, in *2012 28th International Conference on Microelectronics Proceedings*, IEEE, 2012, pp. 371–374.

[101] J. M. Mogollón, H. Guzmán-Miranda, J. Napoles, and M. Aguirre, “Metrics for the measurement of the quality of stimuli in radiation testing using fast hardware emulation”, *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2456–2460, Aug. 2013.

[102] M. Elm and H.-J. Wunderlich, “BISD: Scan-based built-in self-diagnosis”, *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pp. 1243–1248, 2010.

[103] N. Naber, T. Getz, Y. Kim, and J. Petrosky, “Real-time fault detection and diagnostics using FPGA-based architectures”, *Proc. International Conference on Field Programmable Logic and Applications*, pp. 346–351, Aug. 2010.

[104] J. Mogollon, J. Nápoles, H. Guzman-Miranda, and M. Aguirre, “Real time SEU detection and diagnosis for safety or mission-critical ICs using hash library-based fault dictionaries”, *Proc. 2011 12th European Conference on Radiation and Its Effects on Components and Systems*, vol. 3, pp. 705–710, Sep. 2011.

[105] W. Mansour and R. Velazco, “An automated SEU fault-injection method and tool for HDL-based designs”, *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2728–2733, Aug. 2013.

[106] F. Sexton, “Microbeam studies of single-event effects”, *IEEE Transactions on Nuclear Science*, vol. 43, no. 2, pp. 687–695, Apr. 1996.

[107] Y. Xu *et al.*, “An accelerator-based neutron microbeam system for studies of radiation effects”, *Radiation protection dosimetry*, vol. 145, no. 4, pp. 373–376, Jun. 2011.

[108] V. P. Srini, "Special feature: Fault diagnosis of microprocessor systems", *Computer*, vol. 10, no. 1, pp. 60–65, Jan. 1977.

[109] S. Wei, F. Tongshun, and D. Mingfang, "Research for digital circuit fault testing and diagnosis techniques", *Proc. International Conference on Test and Measurement*, vol. 1, pp. 330–333, Dec. 2009.

[110] B. K. Sikdar, N. Ganguly, and P. P. Chaudhuri, "Fault diagnosis of VLSI circuits with cellular automata based pattern classifier", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1115–1131, Jul. 2005.

[111] F. A. Bower, D. J. Sorin, and S. Ozev, "Online diagnosis of hard faults in microprocessors", *ACM Transactions on Architecture and Code Optimization*, vol. 4, no. 2, Jun. 2007.

[112] S. Z. Shazli, M. Abdul-Aziz, M. B. Tahoori, and D. R. Kaeli, "A field analysis of system-level effects of soft errors occurring in microprocessors used in information systems", *Proc. IEEE International Test Conference*, pp. 1–10, Oct. 2008.

[113] P. Bernardi, R. Cantoro, S. De Luca, E. Sanchez, A. Sansonetti, and G. Squillero, "Software-based self-test techniques for dual-issue embedded processors", *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 464–477, 2020.

[114] M. Ulbricht, M. Schölzel, T. Koal, and H. T. Vierhaus, "A new hierarchical built-in self-test with on-chip diagnosis for VLIW processors", *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pp. 143–146, 2011.

[115] D. Kodavade and S. Apte, "Troubleshooting microprocessor based system using an object oriented expert system", *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 5, pp. 111–116, May 2012.

[116] ARM Inc, *CoreSight Technical Introduction*. 2013.

[117] ARM Inc, *CoreSight Architecture Specification v2.0*. 2013.

[118] ARM Inc, *CoreSight Components – Technical Reference Manual*. 2009.

[119] Xilinx Inc, *Zynq-7000 All Programmable SoC: Technical Reference Manual, UG585*. 2016.

[120] ARM Inc, *CoreSight Program Flow Trace Architecture Specification*. 2011.

[121] Xilinx Inc, *Zynq UltraScale+ Device: Technical Reference Manual, UG1085*. 2020.

[122] B. Du, E. Sanchez, M. S. Reorda, J. P. Acle, and A. Tsertov, "FPGA-controlled PCBA power-on self-test using processor's debug features", *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2016.

[123] M. A. Wahab, P. Cotret, M. N. Allah, G. Hiet, V. Lapotre, and G. Gogniat, "ARMHEx: A hardware extension for DIFT on ARM-based SoCs", in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, 2017, pp. 1–7.

[124] Y. Lee, J. Lee, I. Heo, D. Hwang, and Y. Paek, "Using CoreSight PTM to integrate CRA monitoring IPs in an ARM-based SoC", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 3, pp. 1–25, Apr. 2017.

[125] H. Moon, J. Lee, D. Hwang, S. Jung, J. Seo, and Y. Paek, "Architectural supports to protect OS kernels from code-injection attacks and their applications", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 23, no. 1, pp. 1–25, Aug. 2017.

[126] N. Decker *et al.*, "Rapidly adjustable non-intrusive online monitoring for multi-core systems", in *Brazilian Symposium on Formal Methods*, Springer, 2017, pp. 179–196.

[127] N. Decker *et al.*, "Online analysis of debug trace data for embedded systems", in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, pp. 851–856.

[128] D.-A. Suciu and R. Sion, "DroidSentry: Efficient code integrity and control flow verification on TrustZone devices", in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, IEEE, 2017, pp. 156–158.

[129] A. Pârvu, "Program trace capture and analysis for ARM", M.S. thesis, ETH Zürich, Systems Group, Department of Computer Science, 2016.

[130] G. Portokalidis, "TRAILS: Efficient data flow tracking through HW-assisted parallelization", Stevens Institute of Technology Hoboken United States, Tech. Rep., 2019.

[131] E. J. Clark, "Trace-based hardware control flow signature checking", M.S. thesis, University of Illinois at Urbana-Champaign, 2018.

[132] S. M. A. Zeinolabedin, J. Partzsch, and C. Mayr, "Real-time hardware implementation of ARM CoreSight trace decoder", *IEEE Design & Test*, vol. 38, no. 1, pp. 69–77, 2020.

[133] S. M. A. Zeinolabedin, J. Partzsch, and C. Mayr, "Analyzing ARM CoreSight ETMv4.x data trace stream with a real-time hardware accelerator", in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2021, pp. 1606–1609.

[134] A. W. Hoppe, F. L. Kastensmidt, and J. Becker, "Control flow analysis for embedded multi-core hybrid systems", in *Applied Reconfigurable Computing. Architectures Tools and Applications*, Springer, 2018, pp. 485–496.

[135] A. Hoppe, J. Becker, and F. L. Kastensmidt, "Fine grained control flow checking with dedicated FPGA monitors", in *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, IEEE, 2020, pp. 219–224.

[136] M. Glorieux *et al.*, "Single-event characterization of Xilinx Ultrascale+® MPSOC under standard and ultra-high energy heavy-ion irradiation", in *2018 IEEE Radiation Effects Data Workshop (REDW)*, IEEE, 2018, pp. 1–5.

[137] R. Koga, S. Davis, J. George, M. Zakrzewski, and D. Mabry, "Heavy ion and proton induced single event effects on Xilinx Zynq UltraScale+ Field Programmable Gate Array (FPGA)", in *2018 IEEE Radiation Effects Data Workshop (REDW)*, IEEE, 2018, pp. 1–5.

[138] D. S. Lee *et al.*, "Single-event characterization of 16 nm FinFET Xilinx UltraScale+ devices with heavy ion and neutron irradiation", in *2018 IEEE Radiation Effects Data Workshop (REDW)*, IEEE, 2018, pp. 1–8.

[139] M. Amrbar, F. Irom, S. M. Guertin, and G. Allen, "Heavy ion single event effects measurements of Xilinx Zynq-7000 FPGA", in *2015 IEEE Radiation Effects Data Workshop (REDW)*, IEEE, 2015, pp. 1–4.

[140] Digilent Inc, *ZYBO Reference Manual*. 2014.

[141] AVNET Inc, *PicoZed™ 7Z015 / 7Z030 SOM(System-On-Module) Hardware User Guide*. 2020.

[142] J. Gómez-Camacho *et al.*, "Research facilities and highlights at the centro nacional de aceleradores (CNA)", *The European Physical Journal Plus*, vol. 136, no. 3, pp. 1–16, 2021.

[143] R. W. Garnett, "LANSCE accelerator update and future plans", in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1021, 2018, p. 012 001.

[144] Raspberry Pi (Trading) Ltd, *Raspberry Pi 4 Model B datasheet*. 2019.

[145] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, 2011.

[146] T. Michel, R. Leveugle, and G. Saucier, "A new approach to control flow checking without program modification", *Proc. 21th International Symposium on Fault-Tolerant Computing (FTCS-21)*, pp. 334–341, 1991.

[147] A. Lindoso, M. García-Valderas, L. Entrena, Y. Morilla, and P. Martín-Holgado, "Evaluation of the suitability of NEON SIMD microprocessor extensions under proton irradiation", *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1835–1842, Aug. 2018.

[148] M. Pignol, "DMT and DT2: Two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers", *Proc. 12th IEEE International On-Line Testing Symposium*, 10–pp, Jul. 2006.

[149] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith, "Isolation in commodity multicore processors", *Computer*, vol. 40, no. 6, pp. 49–59, Jun. 2007.

[150] Á. B. de Oliveira, G. S. Rodrigues, and F. L. Kastensmidt, "Analyzing lockstep dual-core ARM cortex-A9 soft error mitigation in FreeRTOS applications", *Proc. 30th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pp. 84–89, 2017.

[151] J. R. Azambuja, Â. Lapolli, L. Rosa, and F. L. Kastensmidt, "Detecting SEEs in microprocessors through a non-intrusive hybrid technique", *IEEE Transactions on Nuclear Science*, vol. 58, no. 3, pp. 993–1000, Jun. 2011.

[152] Xilinx Inc, *Soft error mitigation controller v4.1 Product guide, PG036*. 2014.

[153] J. R. Azambuja *et al.*, "Evaluating neutron induced see in SRAM-based FPGA protected by hardware-and software-based fault tolerant techniques", *IEEE Transactions on Nuclear Science*, vol. 60, no. 6, pp. 4243–4250, Dec. 2013.

[154] X. Iturbe, B. Venu, and E. Ozer, "Soft error vulnerability assessment of the real-time safety-related ARM cortex-R5 CPU", *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 91–96, 2016.

[155] N. S. Bowen and D. K. Pradham, "Processor-and memory-based checkpoint and rollback recovery", *Computer*, vol. 26, no. 2, pp. 22–31, Feb. 1993.

[156] Á. B. de Oliveira *et al.*, "Lockstep dual-core ARM A9: Implementation and resilience analysis under heavy ion-induced soft errors", *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1783–1790, Aug. 2018.

[157] M. Violante, C. Meinhardt, R. Reis, and M. S. Reorda, "A low-cost solution for deploying processor cores in harsh environments", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 2617–2626, Jul. 2011.

[158] V. Bernon-Enjalbert, M. Blazy-Winning, R. Gubian, D. Lopez, J.-P. Meunier, and M. O'Donnell, "Safety integrated hardware solutions to support ASIL D applications", 2013.

[159] A. Shye, J. Blomstedt, T. Moseley, V. J. Reddi, and D. A. Connors, "PLR: A software approach to transient fault tolerance for multicore architectures", *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 2, pp. 135–148, Apr. 2009.

[160] H. Mushtaq, Z. Al-Ars, and K. Bertels, "Efficient software-based fault tolerance approach on multicore platforms", *Design, Automation & Test in Europe Conference & Exhibition*, pp. 921–926, 2013.

[161] G. S. Rodrigues, F. Rosa, Á. B. de Oliveira, F. L. Kastensmidt, L. Ost, and R. Reis, "Analyzing the impact of fault-tolerance methods in ARM processors under soft errors running Linux and parallelization APIs", *IEEE Transactions on Nuclear Science*, vol. 64, no. 8, pp. 2196–2203, Aug. 2017.

[162] ARM Inc, *Cortex-A9 MPCore Technical Reference Manual*. 2011.

[163] A. Serrano-Cases, F. Restrepo-Calle, S. Cuenca-Asensi, and A. Martínez-Álvarez, "Softerror mitigation for multi-core processors based on thread replication", *Proceedings of 20th IEEE Latin American Test Symposium*, Mar. 2019.

[164] G. M. Swift, F. Fannanesh, S. M. Guertin, F. Irom, and D. G. Millward, "Single-event upset in the PowerPC750 microprocessor", *IEEE Transactions on Nuclear Science*, vol. 48, no. 6, pp. 1822–1827, Dec. 2001.

[165] H. Asadi, V. Sridharan, M. B. Tahoori, and D. Kaeli, "Vulnerability analysis of l2 cache elements to single event upsets", *Proc. of the Design Automation & Test in Europe Conference*, pp. 1276–1281, Mar. 2006.

[166] ARM Inc, *Cortex-A9 Technical Reference Manual r4p1*. 2012.

[167] A. Lindoso, M. Garcia-Valderas, and L. Entrena, "Analysis of neutron sensitivity and data-flow error detection in ARM microprocessors using NEON SIMD extensions", *Microelectronics Reliability*, vol. 100, no. 113346, 2019.

[168] ARM Inc, *Cortex-A9 NEON Media Processing Engine Technical Reference Manual r3p0*. 2011.

[169] ARM Inc, *NEON Programmer's Guide*. 2013.

[170] M. Pignol, "COTS-based applications in space avionics", *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pp. 1213–1219, 2010.

[171] L. Entrena, A. Lindoso, M. G. Valderas, M. Portela, and C. L. Ongil, "Analysis of SET effects in a PIC microprocessor for selective hardening", *IEEE Transactions on Nuclear Science*, vol. 58, no. 3, pp. 1078–1085, Feb. 2011.