Universidad Carlos III de Madrid
www.uc3m.es

TESIS DOCTORAL

Nueva metodología para el endurecimiento óptimo de sistemas digitales con distribución de la funcionalidad, trabajando en entornos sometidos a la radiación ionizante

Autora:
Anna Vaskova

Directora:
Celia López Ongil

DEPARTAMENTO TÉCNOLOGÍA ELECTRÓNICA

UNIVERSIDAD CARLOS III DE MADRID

Leganés, JUNIO 2016

Universidad Carlos III de Madrid
www.uc3m.es

TESIS DOCTORAL

Nueva metodología para el endurecimiento óptimo de sistemas digitales con distribución de la funcionalidad, trabajando en entornos sometidos a la radiación ionizante

Autora: *Anna Vaskova*

Directora: **Celia López Ongil**

Firma del Tribunal Calificador:

Firma

Presidente:

Vocal:

Secretario:

Calificación:

Leganés, de de

Quiero dedicar este trabajo a todos los profesores que he tenido a lo largo de mi carrera de estudiante. He tenido mucha suerte aprender de sus clases y también de la vida.
A Marina y a Yuri.

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento a mi directora de tesis, Celia López, por todo el trabajo, esfuerzo y dedicación que ha puesto para el desarrollo de este trabajo. Trabajo el cual ha sido desarrollado en el marco de los proyectos financiados por el Ministerio de Economía y Competitividad.

Mi agradecimiento a todos los miembros que forman parte del grupo de Diseño Microelectrónico y Aplicaciones de la UC3M por sus consejos y el apoyo durante estos años, en especial a Marta Portela García y a Mario García Valderas que me han sido de mucha ayuda a lo largo de la investigación.

También agradecerle al grupo de investigación del CNA (Centro Nacional de Aceleradores) por cedernos el uso de sus instalaciones para irradiación de los dispositivos. En especial, mi agradecimiento a Yolanda Morilla, a Gema Muñiz y Mª Carmen Jiménez por su ayuda inestimable en la realización de los tests y en el análisis de los datos.

Muchas gracias a Matteo Sonza Reorda del Politecnico di Torino y a su grupo de investigación, por su colaboración en la investigación.

Gracias al Departamento de Ciencia e Ingeniería de los Materiales e Ingeniería Química de la UC3M que nos ayudó y realizó el estudio de las composiciones del material a radiar con el microscopio electrónico de barrido.

También quería agradecer INTA (Instituto Nacional de Técnica Aeroespacial) y en especial a Alberto Martín Ortega, por darnos todo su apoyo y por ser como es.

Resumen

Debido al extenso uso de los sistemas digitales distribuidos en las diferentes áreas tecnológicas sometidas a entornos agresivos, como por ejemplo la automoción o el espacio, actualmente es de vital importancia la aplicación de técnicas de endurecimiento ante los efectos de radiación ionizante. Esta tesis se inició con el proyecto de investigación RENASER+ "TEC2010-22095-C03-03", financiado por el Ministerio Español de Ciencia y Tecnología. El objetivo principal del subproyecto liderado por la Universidad Carlos III de Madrid, consistía en la evaluación de la sensibilidad de los sistemas digitales a los efectos de la radiación ionizante. Para ello, se planificaron una serie de tareas destinadas a establecer un protocolo y desarrollar un sistema de pruebas válido para realizar campañas de irradiación en los aceleradores instalados en el Centro Nacional de Aceleradores de la Universidad de Sevilla (CNA). Con el fin de alcanzar los objetivos de este proyecto, en los trabajos de la tesis doctoral, se estudiaron y aplicaron técnicas de tolerancia a fallos para sistemas distribuidos, y se comprobó la sensibilidad del sistema con el método basado en la inyección de los fallos por emulación. En concreto, utilizando la herramienta de Emulación Autónoma desarrollada en el grupo DMA de la Universidad Carlos III. A lo largo del desarrollo de la tesis doctoral, en concordancia con las tareas y objetivos del proyecto RENASER+, se han introducido mejoras en el sistema de inyección de fallos transitorios por emulación, para permitir la inyección de fallos múltiples en los elementos de memoria y para permitir el análisis detallado del efecto acumulado de los fallos detectados mediante técnicas de mitigación de fallos, y técnicas de rastreo del fallo y su propagación. Con el objetivo de generar un método integral de análisis, validación y calificación de los sistemas distribuidos robustos, además de inyectar fallos por emulación se indujeron fallos reales en dichos sistemas, mediante el uso del acelerador de partículas tipo ciclotrón del CNA de Sevilla. La evaluación de la sensibilidad del sistema distribuido prototipado requirió una cualificación mediante varias campañas de irradiación, en esta tesis se ha propuesto un método de planificación y realización de una campaña que permite la monitorización continúa del efecto de los fallos y la generación automática de informes finales del proceso completo. En la realización de la tesis doctoral se planteó un objetivo más ambicioso, que consistía en la generación de un método de comprobación continua, durante el tiempo de funcionamiento normal de los sistemas distribuidos, de la presencia de fallos que pudieran causar un mal funcionamiento, e incluso averías graves. Así, se amplió el método de calificación de la robustez de sistemas distribuidos a un método de test *on-line* de dichos sistemas. Durante el estudio de la robustez de los sistemas distribuidos, principalmente para aplicaciones terrestres, se identificó y estudió el problema de los fallos debidos a envejecimiento de los componentes electrónicos digitales. Por lo tanto, el método propuesto es también adecuado para la realización del *test on-line* durante el funcionamiento normal del sistema distribuido, para la detección de los fallos procedentes de la radiación ionizante (fallos transitorios) y los fallos debidos al envejecimiento de los dispositivos (fallos permanentes).

Los sistemas digitales distribuidos, aparte de los nodos que los componen, tienen un protocolo de comunicación que puede fallar igual que el mismo nodo. En esta tesis doctoral se han estudiado varios protocolos de comunicación utilizados en sistemas distribuidos de aplicaciones aeroespaciales y de automoción; así mismo, se han comparado los protocolos CAN y LIN, y se ha propuesto, diseñado y prototipado un sistema distribuido básico, para la comprobación de la metodología de calificación y *test on-line* y de las técnicas de endurecimiento propuestas.

Como aportación original de esta tesis doctoral, se propone una nueva metodología para comprobar y asegurar el endurecimiento efectivo de un sistema digital distribuido, que incluya cualquier bus de comunicación. Otra contribución original de esta tesis doctoral, consiste en una técnica que permite la comparación de los datos obtenidos tras la irradiación de una forma directa y transparente al sistema de *test*, basada en un bloque de detección de minoría para rastreo de la propagación de los fallos. Se han analizado distintas tecnologías de FPGA basadas en memoria Flash, con el propósito de una mejor caracterización de los dispositivos a iradiar. Para el sistema final se ha optado por la utilización de una FPGA Igloo de Microsemi®, debido a que es una tecnología más robusta y basada en la tecnología Flash. Se ha desarrollado un software de control para el sistema distribuido, el cual se ejecuta en un microprocesador contenido en uno de los nodos del sistema y el que envía los resultados obtenidos por medio de un bus SPI a un PC. El software automatiza el proceso de recolección de los datos, proporcionando el resultado de cuál de los nodos ha fallado, si se ha recuperado, si ha fallado el sistema de comunicación, etc. Estos resultados permiten validar experimentalmente el método propuesto para los sistemas distribuidos digitales. Además de cumplir con los objetivos del proyecto, se han resuelto dos de los problemas típicos de este tipo de sistemas. Estos corresponden al *reset* síncrono del sistema y a la monitorización del mantenimiento continuo del tiempo real del sistema. Los distintos bloques propuestos y utilizados en este método integral de calificación de sistemas distribuidos son adecuados para la realización de test *on-line* de sistemas distribuidos, con el objeto de detectar la presencia de fallos permanentes (debidos al envejecimiento de los dispositivos) y de fallos transitorios (debidos a la radiación ionizante).

Finalmente, los últimos aportes de la investigación de este trabajo de tesis, correspondientes al análisis detallado del material del encapsulado del dispositivo a irradiar, y su comportamiento ante el paso de protones, se han podido obtener gracias a la ayuda del Departamento de Ciencia e Ingeniería de los Materiales e Ingeniería Química de la Universidad Carlos III de Madrid y del Centro Nacional de Aceleradores de la Universidad de Sevilla.

La mayoría de los resultados parciales y globales de este trabajo de tesis doctoral han sido publicados en Conferencias Internacionales (International On-Line Testing Symposium, Latin American Test Workshop, Digital Circuit and Integrated Systems) y en una revista internacional (IEEE Transactions on Nuclear Science).

Abstract

Nowadays, due to the spread of distributed digital systems in the different technological areas, working in aggressive environments, such as automotive and aerospace applications, hardening techniques against ionizing radiation effects is crucial. This PhD work started with the research Project RENASER+ "TEC2010-22095-C03-03", supported by the Spanish Ministry of Economy and Competitiveness. The main objective of the sub-project, managed by the Carlos III University of Madrid, was the assessment of the sensitiveness of digital systems against ionizing radiation effects. For this purpose, a number of tasks were planned, aimed to set a protocol and to develop a test system, able to run irradiation campaigns in the accelerators placed in the National Centre of Accelerators, University of Seville (CNA). Furthermore, in order to fulfill the project objectives, the works developed in the PhD were devoted to the study and application of Fault Tolerance Techniques for distributed digital systems, and to the prior analysis of the systems sensitivity with emulation-based fault injection campaigns. This last task used the Autonomous Emulation Tool developed in the DMA research group of Carlos III University of Madrid. Along the development of the work of this PdD, in agreement with the tasks and objectives of RENASER+ Project, new improvements have been proposed, developed and included in the emulation-based transient fault injection tool, in order to enable the injection of multiple faults in memory elements, to enable the detailed tracking of accumulated effect of faults injected and detected by mitigation techniques and to scan the fault propagation within the distributed system. With the main purpose of generating a comprehensive method for the analysis, assessment and qualification of robust digital distributed systems, apart from injecting faults via emulation, real faults have been injected thanks to the use of a particle accelerators, CNA-US cyclotron (proton beam). The assessment of the prototyped distributed digital systems, through proton beam irradiation campaigns, required three irradiation campaigns; in the PhD a planning method was proposed for this type of campaigns on distributed systems, which allows the continuous monitoring of fault effects and automatic final report generation. As a more ambitious objective, in the PhD the method was extended to an *on-line* test, for the continuous checking during normal operation of distributed digital systems, to detect faults which can cause a wrong behavior or, even, a serious failure. Therefore, the qualification method for the assessment of the robustness of distributed digital systems was extended to an *on-line* method for these systems. During the robustness study of distributed digital systems another problem was identified and analyzed: the permanent faults appearing in this type of systems due to digital devices aging. Indeed, the proposed method is suitable for *on-line* testing of the normal operation of the distributed digital systems, for the detection of faults due to ionizing radiation (transient faults) and the detection of faults due to the device aging (permanent faults).

Distributed digital systems, apart the nodes composing them, use to include a communication protocol that can suffer also from transient or permanent faults. In this PhD different communication protocols, used in distributed digital systems for

aerospace and automotive applications; also, CAN and LIN protocols have been compared and a basic distributed system has been proposed, designed and prototyped, for validating the qualification methodology and the *test on-line* method, as well as proposed hardening techniques.

As original contribution of this PhD, a comprehensive methodology for assessing and assuring the effective hardening of distributed digital systems, including a communication protocol, is proposed. Another original contribution of this PhD is the technique that allows the analysis of the data obtained (transparently and directly whatever the test system used) from any irradiation campaign, based on a minority checker block for the fault tracking. Furthermore, in this PhD work, different Flash-based FPGA technologies have been analyzed, with the purpose of better characterizing the analyzed devices. For the final system implemented, FPGAs Igloo from Microsemi® have been selected, because is a more robust technology. Specific control software has been developed for the distributed system, which was run on a microprocessor included in one node of the system and that sends obtained results through a PCI bus to a Personal Computer. This software automates the data collecting process, telling which node is failing, its recovery state, as well as any communication fault occurring. These tools allow the experimental validation of proposed method for the distributed digital systems. Apart from fulfilling the Project objectives, two typical problems of this type of systems have been solved. They correspond to synchronous initialization of the system and continuous real-time maintenance. Different blocks proposed and used in this comprehensive method for the assessment of the sensitivity of distributed digital systems are also adequate for *on-line* testing of these systems, with the prupose of detecting permanent faults (due to device aging) and transient faults (due to ionizing radiation).

Finally, the latest research works of this PhD work correspond to the detailed analysis of the packaging of the devices to be irradiated, as well as its behavior when proton particles are going through it. These works have been done thanks to the help of Science and Material Engineering and Chemistry Engineering Department of Carlos III University of Madrid, and the National Centre of Accelerators of University of Seville.

The majority of partial and global results of this PhD have been published in International Conferences (International On-Line Testing Symposium, Latin American Test Workshop, Digital Circuit and Integrated Systems) and in the IEEE Transactions on Nuclear Science.

ÍNDICE

Índice de Tablas

Índice de Figuras

Acrónimos

ASIC	***Application Specific Integrated Circuit***
ASIL	***Automotive Safety Integrity Level***
ABS	***Antilock Brake System***
ATV	***Automated Transfer Vehicle***
CAD	***Computer Aided Design***
CAN	***Controller Area Network***
CRC	***Cyclic Redundancy Check***
CI	***Circuito Integrado***
CNA	***Centro Nacional de Aceleraciones***
CNES	***Centre National D'Etudes Spatiales***
COTS	***Commercial Off-the-Shelf Technologies***
CPU	***Central Processing Unit***
DD	***Displacement Damage***
DFT	***Design for Testability***
ECU	***Engine Control Unit***
ECC	***Error Correction Codes***
EDC	***Error Detection Codes***
EMI	***Electromagnetic Interference***
ESA	***European Space Agency***
FIT	***Fault Isolation and Test***
FUT	***Functionally Untestable***
FPGA	***Field Programmable Gate Array***
HSSL	***High Speed Serial Link***
IEC	***International Electrotechnical Commission***
ISO	***International Organization for Standardization***
JEDEC	***Joint Electron Device Engineering Council***
JESD	***JEDEC standard***
LEO	***Low Earth Orbit***
LET	***Linear Energy Transfer***
LIN	***Local Interconnect Network***
LUT	***Look-Up Table***
MISR	***Multiple Input Shift Register***
NASA	***National Aeronautics and Space Administration***
NIEL	***Non-Ionizing Energy Loss***
NMR	***N-Modular Redundancy***
OCD	***On-Chip Debugger***
RRTC	***Robust Real Time Counter***
SCC	***Space Components Coordination***
SE	***Soft Error***
SEE	***Single Event Effect***
SEFI	***Single Event Functional Interrupt***
SEL	***Single Event Latch-up***
SEM	***Scanning Electron Microscope***
SER	***Soft Error Rate***
SET	***Single Event Transient***
SEU	***Single Event Upset***
SoC	***System on Chip***
SoPC	***System on Programmable Chip***

SRAM	***Static Random Access Memory***
TID	***Total Ionizing Dose***
TMR	***Triple Modular Redundancy***
VHDL	***VHSIC Hardware Description Language***
VHSIC	***Very High Speed Integrated Circuit***
VLSI	***Very Large Scale Integration***

Capítulo 1 INTRODUCCIÓN

Hoy en día, hay cada vez más sistemas electrónicos en los sectores de automoción, aeronáutica, redes de sensores, *Internet of Things*, *Cyberphysical Systems*, etc. Así mismo, los sistemas electrónicos digitales se encuentran en cualquier sector que incluya el control electrónico. Hay un gran número de tipos diferentes de sistemas digitales que se usan en diversos entornos relacionados con el campo de la industria (en el control industrial automático, robots, vehículos), las telecomunicaciones (teleoperaciones, circuitos de vigilancia), el campo aeroespacial (satélites, controladores aéreos) o el biomédico (circuitos de procesamiento digital de datos, modelado de comportamientos fisiológicos, etc.), y cada día está creciendo la complejidad y la diversidad de estos sistemas. Debido a esta creciente utilización de sistemas digitales construidos con tecnología de dimensiones nanométricas, los efectos de posibles fallos, procedentes del entorno de trabajo o bien del envejecimiento del circuito, son cada vez más perjudiciales y abundantes.

Actualmente, los sistemas electrónicos se apoyan en gran medida en la distribución de tareas, con el fin de utilizar componentes de complejidad media, en lugar de dispositivos de altas prestaciones computacionales, para repartir eficientemente las tareas y para aumentar la fiabilidad del conjunto. El proceso de diseño y desarrollo implica, necesariamente, una especial y cuidadosa atención a la fiabilidad y la seguridad de la implementación de estos circuitos o sistemas digitales.

Inicialmente, los sistemas distribuidos han tenido interés y han sido investigados en relación con las aplicaciones informáticas. Los primeros sistemas operativos eran sistemas de procesamiento por lotes (en inglés *batch processing*), que permitían procesar en diferido, sin el control por el usuario y enviando secuencialmente datos suministrados en paquetes. Hoy en día, el procesamiento por lotes se aplica en aplicaciones de cálculos intensivos, por ejemplo en las aplicaciones de supercomputación. Los sistemas centralizados fueron la continuación mejorada de los sistemas de procesamiento por lotes. Con aquel tipo de sistemas se logró incrementar la eficiencia en el uso de la unidad central de procesamiento (CPU) de los microprocesadores, un recurso entonces caro y escaso, y disminuir los tiempos de respuesta para los usuarios. Los sistemas en red aparecen cuando los terminales con computadoras personales fueron ganando capacidad de cálculo y funcionalidad hasta convertirse en sistemas autónomos. Es entonces cuando aparecen los sistemas distribuidos, que constan de un conjunto de computadores que se interconectan. La diferencia fundamental con los sistemas en red es que la ubicación del dispositivo es transparente a las aplicaciones y usuarios. El usuario accede a los recursos del sistema distribuido a través de una interfaz gráfica desde un terminal, despreocupándose de su localización. Las aplicaciones ejecutan una interfaz de llamadas al sistema como en los sistemas centralizados. Los sistemas distribuidos proporcionan de forma transparente los recursos compartidos, facilitando el acceso y la gestión, e incrementando la eficiencia y la disponibilidad [1].

Con el progreso de la sociedad, el campo de la utilización de los sistemas distribuidos ha pasado al control digital en numerosas aplicaciones, mediante

dispositivos reconfigurables (FPGAs) o programables (Microprocesadores) de propósito específico. Un sistema distribuido puede estar compuesto por varios nodos que operan con o sin baterías, y que se distribuyen a lo largo de un entorno de interés particular. Cada nodo en la red recolecta datos de su ambiente, o realiza las tareas específicas de su entorno. Cada nodo sensor, puede enviar los datos recolectados para que la información sea procesada por una computadora.

El gran avance de los sistemas distribuidos digitales facilita el traspaso de los conceptos fundamentales de los sistemas de red informática al área de control digital. Es común encontrar los sistemas distribuidos digitales en sistemas construidos con componentes o bloques empotrados donde diversos sensores, actuadores y dispositivos de control, con diferentes niveles de complejidad, se encuentran dispersos. Con estos sistemas se consigue no sólo una disminución del cableado sino también una reducción de ancho de banda requerido, a la vez que se mejora la fiabilidad y controlabilidad del sistema.

Actualmente, en la literatura existen pocas soluciones para proteger los sistemas distribuidos, que mayoritariamente están sometidos a entornos muy críticos, frente a fallos que pueden afectarles. La fiabilidad en este tipo de sistemas es un factor crucial dado el carácter autónomo e intensivo en transmisión de datos y comandos. Para poder analizar cómo aumentar la fiabilidad de estos sistemas distribuidos, en primer lugar definimos qué tipos de fallos pueden afectar a un sistema distribuido digital.

Uno de los factores que puede afectar al comportamiento normal de un circuito provocando un fallo son las condiciones ambientales, como por ejemplo variaciones de la temperatura, interferencias electromagnéticas (EMI), y en especial la radiación ionizante. El gran desarrollo de las tecnologías micro y nano electrónicas ha provocado un aumento significativo de la sensibilidad a dichos factores. Se han reducido las dimensiones de los transistores hasta escalas nanométricas (Intel anuncia en *International Solid-State Circuits Conference* (ISSCC 2015) fabricación de los chips con transistores de 7 nm para el año 2017) y las frecuencias de funcionamiento se han incrementado siendo los circuitos cada vez más rápidos (con frecuencias de ~100GHz). Teniendo esto en cuenta, hay que enfrentarse a los problemas de pérdida de la confiabilidad y aumento de la sensibilidad del sistema.

Los fallos más críticos de los sistemas digitales son debidos a la radiación ionizante. La radiación puede ser no ionizante e ionizante. La radiación no ionizante es la que no tiene suficiente energía para ionizar los átomos de material al contactar con él, los ejemplos de las fuentes de este tipo de radiación son las microondas, la luz visible, las ondas de radio (ondas electromagnéticas de frecuencias inferiores a 300 GHz). Aunque la energía de este tipo de radiación es débil para romper enlaces químicos, pueden producir los efectos leves biológicos que son el calentamiento y la inducción de corrientes eléctricas en los tejidos y células. Los efectos sobre la salud humana es un aspecto polémico en el campo de la ciencia, pero no hay efectos sobre los circuitos electrónicos digitales.

La radiación ionizante tiene la energía suficiente para provocar un efecto en el material. Es la que se debe a las partículas alfa, beta, neutrones, protones, iones pesados. Algunas de las fuentes externas de radiación ionizante son los rayos cósmicos, rayos gamma, rayos X, irradiación o viento solar, partículas interestelares. El tipo y la cantidad de estas partículas ionizantes varían según la localización del sistema digital (espacio, órbitas terrestres, aviónica, etc.). El flujo de radiación cósmica depende de la altitud y la latitud sobre la superficie terrestre, siendo menor al disminuir aquella y al aumentar esta. Los rayos cósmicos generan partículas cargadas de gran energía que provocan los *Single Event Effect* (SEE) en los circuitos digitales. Estos efectos pueden

provocar un mal funcionamiento de las aplicaciones o incluso causar un daño permanente en los circuitos afectados.

En general, las partículas ionizantes no llegan a la atmósfera terrestre gracias a la protección de la magnetosfera. Cuando dicha radiación atraviesa la atmósfera terrestre los protones y electrones, procedentes del viento solar y otras fuentes externas, reaccionan con las partículas de la atmósfera generando cascadas de partículas secundarias, Figura 1, Figura 2. Las partículas secundarias con suficiente energía para continuar atravesando la atmósfera interaccionan a su vez generando nuevas partículas, y así sucesivamente hasta que algunas partículas (menos del 1% del flujo de partículas inicial pero siendo bastantes para poder afectar a los materiales) alcanzan la superficie terrestre. Dentro del flujo de radiación secundaria que alcanza la superficie terrestre los neutrones son uno de los componentes más numerosos. Los neutrones son partículas sin carga por lo que no generan directamente la ionización de un material, como el silicio en los circuitos integrados pero, debido a su masa, generan iones de alta energía que sí provocan la aparición de pares electrón-hueco dando lugar a un pulso de corriente. Por ejemplo en área terrestre de Nueva York, caen de orden de 13 neutrones por centímetro cuadrado cada segundo[3].

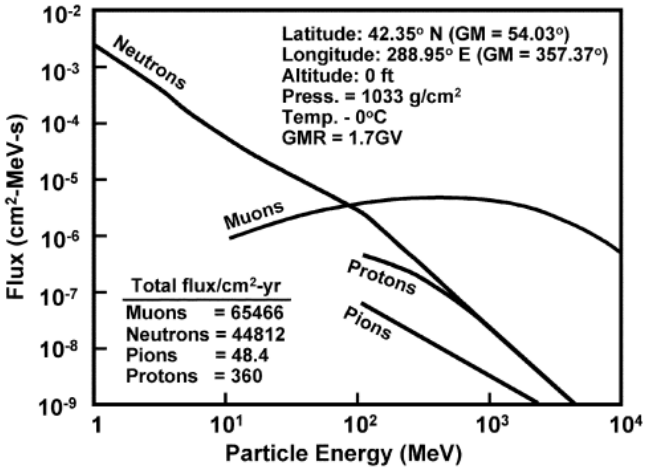


Figura 1. Fuentes de radiación cósmica a nivel de tierra en Nueva York [48].

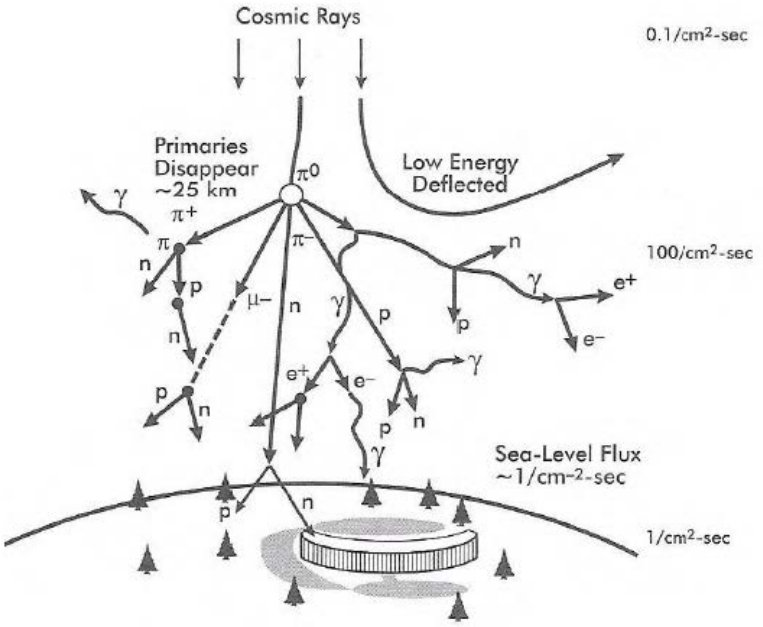


Figura 2. Partículas procedentes de los rayos cósmicos [48].

Además, en la superficie terrestre, existe una fuente de radiación natural producida por las partículas alfa de los materiales radioactivos [2]. En resumen, se observa que las fuentes de radiación ionizante que afectan a los sistemas electrónicos tienen más efecto en el exterior de la atmósfera terrestre y una vez dentro de esta en alturas más elevadas.

La energía necesaria para producir la ionización en un elemento depende de su número atómico. En los elementos ligeros es del orden de decenas de eV, para aire se acepta el valor de 34 eV. Aunque no toda la energía se va a ionizar, una radiación de energía de varios MeV es capaz de producir un total de unos 100 000 pares ión-electrón en el aire. El efecto que produce la ionización es distinto para cada tipo de radiación, su energía y el material a que afecta.

Por lo tanto, las aplicaciones aeroespaciales son las más sensibles, al funcionar expuestas a la radiación cósmica primaría, por lo que el estudio y la prevención de los efectos de la ionización es un problema fundamental en dicho ámbito. Por eso, tradicionalmente los SEE eran un problema únicamente en las aplicaciones aeroespaciales. Sin embargo, debido a las características de la tecnología actual también la radiación que llega a la superficie terrestre, principalmente neutrones, provoca fallos en los circuitos digitales que funcionan a nivel terrestre, puesto que la corriente generada por la interacción es suficiente para generar un fallo. Como consecuencia, los efectos de la radiación se han generalizado, planteando un problema de gran dificultad para las aplicaciones más críticas, tales como las de automoción, medicina, control del tráfico, aviónica, etc. El efecto de la radiación en las tecnologías actuales está considerado como un factor a resolver para continuar con el progreso tecnológico [6]. Los circuitos electrónicos simples se han estudiado mucho, tanto su endurecimiento como la detección on-line de fallos, sin embargo la robustez de los sistemas distribuidos frente a radiación ionizante no está aún resuelta.

Entre las soluciones propuestas para el endurecimiento de tales aplicaciones están las técnicas tradicionales de tolerancia de fallos y los nuevos métodos específicos que se benefician de algunas características del sistema. Estudiando los trabajos realizados en este campo, actualmente no está disponible ningún método para evaluar de forma eficiente y fiable la robustez de los sistemas electrónicos distribuidos en aplicaciones con tareas críticas. Estas soluciones deben estar dirigidas a las implementaciones en hardware, con baja potencia del consumo, alta velocidad de procesamiento y peso mínimo lo que las hace ideales para aplicaciones con tareas críticas y espaciales.

1.1. MOTIVACIÓN

Debido al continuo crecimiento de la complejidad de los sistemas con circuitos digitales que tienen tareas distribuidas, el estudio detallado de todo el sistema es necesario. En la mayoría de las aplicaciones es crucial tener uno o varios mecanismos para garantizar el nivel requerido de la fiabilidad del sistema, aunque esto sea dependiente del campo de la aplicación. La variedad de los distintos tipos de efectos relacionados con el envejecimiento, el desgaste o la radiación, aumenta la necesidad de soluciones a nivel de sistema tanto previamente a la aparición de los fallos o una vez que estos se hayan producido. El actual interés en producir tests *on-line* que resulten económicos, eficientes y precisos es la principal motivación de esta tesis.

1.2. OBJETIVOS

El objetivo principal se va a centrar en el endurecimiento óptimo de los distintos componentes de un sistema distribuido. Como punto de partida se realizará un estudio de los diferentes dispositivos existentes hoy en día en el mercado utilizados para construir este tipo de sistemas. Se realizará una selección de los componentes suficientemente robustos ante la radiación ionizante. A continuación se hará el

endurecimiento parcial de los circuitos combinacionales y secuenciales mediante el análisis de las probabilidades del enmascaramiento de los fallos. Por otro lado, se hará un endurecimiento de los circuitos integrados digitales mediante la adición de las redundancias al nivel más bajo, para evitar el efecto de los eventos transitorios. Una vez realizados los pasos previos, se realizará el endurecimiento completo de los sistemas distribuidos. Para ello se hará la definición de las tareas críticas comunes a distribuir entre los distintos nodos del sistema. Se realizará la selección de los nodos básicos que va a presentar el sistema distribuido, que incluirá entre otros un microprocesador embebido. Además el endurecimiento del sistema se realizará mediante el reparto de las tareas críticas entre el hardware y el software en un mismo nodo y a nivel de sistema. Una vez realizado el punto anterior se realizará la aplicación y la evaluación de las técnicas utilizadas del endurecimiento colaborativo.

Otro de los objetivos consistirá en la evaluación de una plataforma para el análisis integral de la sensibilidad de dispositivos embarcados. Esta evaluación se realizará mediante el estudio de la sensibilidad de los componentes utilizados por medio de la técnica de emulación autónoma. Se hará un análisis detallado del comportamiento de los fallos detectados mediante las técnicas de mitigación, para evaluar que fallos pueden convertirse en averías posteriores si el sistema no responde adecuadamente. También se verificará la sensibilidad de los sistemas distribuidos mediante la técnica de emulación autónoma. Para este fin se va a realizar el análisis y el endurecimiento de los protocolos de comunicación de las redes CAN y LIN usadas ampliamente hoy en día en los sectores del automóvil y el aeroespacial. Se propondrán y se desarrollarán las técnicas de endurecimiento para los dispositivos utilizados en los sistemas citados anteriormente.

Finalmente se elaborará un procedimiento para los ensayos de pre-certificación por irradiación destinado a los sistemas digitales con capacidad de diagnóstico. Se llevarán al cabo varias campañas de irradiación en las instalaciones de CNA. Se realizará una campaña de validación de los dispositivos CoolRunner-II ™ de Xilinx. También se hará otra campaña de calificación de los dispositivos programables Igloo™ de Microsemi. Y como punto final se harán varias campañas de calificación de los sistemas distribuidos completos.

1.3. ORGANIZACIÓN

El documento de esta tesis doctoral está dividido en seis capítulos. El primer capítulo introduce los conceptos generales relacionados con el efecto de las distintas partículas sobre los dispositivos electrónicos, el segundo capítulo describe los tipos y los efectos de los fallos y los métodos para su evaluación a nivel de circuito y a nivel de sistema. En los capítulos tres y cuatro se propone un sistema distribuido estándar en el cual se analiza la sensibilidad de los distintos protocolos de comunicación ante fallos transitorios y permanentes. El quinto capítulo describe los resultados experimentales obtenidos y finalmente, en el capítulo sexto se exponen las conclusiones obtenidas y las líneas futuras de trabajo a seguir.

Capítulo 2 ESTUDIO DE LA TÉCNICA

Cuando un sistema electrónico digital se está utilizando en un entorno crítico en seguridad como las aplicaciones en automoción, espacio o aviónica, los reglamentos y las normas pueden exigir unos niveles mínimos de tolerancia a la presencia de fallos.

Hasta hace pocos años, la robustez en los sistemas electrónicos no era un aspecto muy crítico, debido a que había muy pocos equipos electrónicos en las aplicaciones terrestres y no era necesario que fuesen funcionalmente seguros, puesto que no suponían riesgos para la población o infraestructuras. Pero hoy en día, hay numerosos circuitos electrónicos en casi todos los niveles, aplicaciones de control, seguridad, sensores, bienestar, etc., resultando muy importante elaborar estándares para asegurar la fiabilidad de los circuitos electrónicos en distintos entornos, considerando los fallos que puedan afectarles. Por ejemplo, un sistema de control dinámico de la estabilidad (ESP) o un sistema de distribución de la fuerza de frenado (EBV) al fallar puede tener consecuencias fatales. De ahí la obligación de cumplir con las normas definidas en los estándares, si están definidos en la aplicación que se considera, o de definir estándares realistas y sencillos.

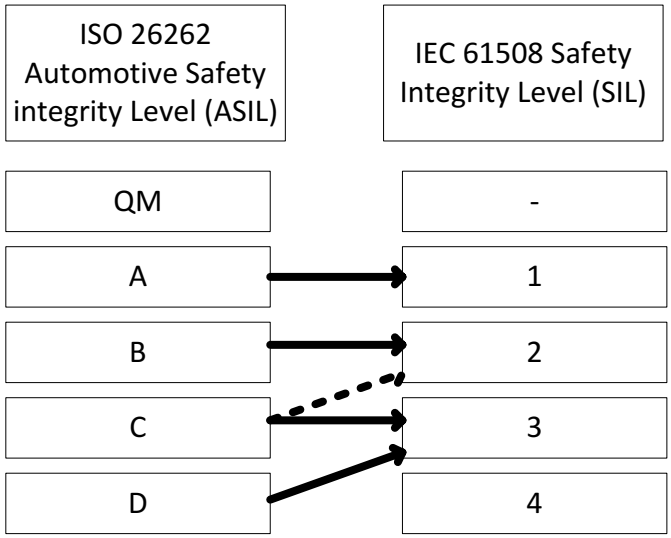


Figura 3. Correlación entre las normas SIL y ASIL [78]

Hay numerosos ejemplos de este intento de normalizar el desarrollo y la validación de los sistemas electrónicos digitales que están incluidos en aplicaciones críticas. Por ejemplo, la Comisión Internacional Electrotécnica (IEC), define la norma IEC 61508 para la seguridad funcional de sistemas eléctricos, electrónicos y electrónicos programables relacionados con la seguridad. Por otra parte, la norma ISO 26262 se deriva de la norma IEC 61508, siendo ISO 26262 la norma que fija los niveles de seguridad en la industria de la automoción ASIL (*Automotive Safety Integrity Level*) [78], y que acaba de entrar en vigor. Los desarrolladores de sistemas de automoción deben asegurarse de que sus productos cumplen con sus especificaciones. Las dos normas definen cuatro niveles de seguridad integral (SIL). En función de la evaluación

del análisis de riesgos, SIL1 (A) se considera como el más bajo nivel y SIL4 (D) el más alto.

Para sistemas que trabajan en aplicaciones aeroespaciales las especificaciones son aún más estrictas. La agencia espacial europea (ESA) y la agencia espacial norteamericana (NASA) están utilizando en la estación espacial internacional (ISS *International Space Station*) circuitos electrónicos endurecidos frente a la radiación aplicando varias normas. En estos estándares se especifican los requisitos de las pruebas a realizar para calificar los equipos y sus componentes, características de las fuentes de irradiación, dosimetría, etc. Entre ellos, por ejemplo, las normas ESA SCC *Basic Specification* 22900 y ESA SCC (*Space Components Coordination*) *Basic Specification 25100* son los principales estándares para la realización de los *tests* de irradiación con protones; además, la norma *NASA* SSP 30512 define el entorno de radiación ionizante en la estación espacial para el diseño de los equipos electrónicos y cargas útiles (*payloads*) que se utilicen, así como las pruebas de exposición para la tripulación. Finalmente, JEDEC JESD57 [11] es un estándar global para realizar los *tests* de irradiación de circuitos microelectrónicos que entre las otras agencias utiliza la NASA [7]. En la Tabla 1 se enumeran los estándares más utilizados en el campo de la calificación de los circuitos electrónicos para aplicaciones aeroespaciales.

Estándar	Titulo	Data
JEDEC JESD57	Procedimiento de *test* para la medición de los fallos transitorios (SEEs) producidos en los dispositivos semiconductores por irradiación de los iones pesados. *Test Procedures for the Measurement of SEE in Semiconductor Devices from Heavy-Ion Irradiation.*	1996
JEDEC JESD234	Estándar para la medición de los fallos transitorios SEEs en dispositivos electrónicos por irradiación con protones. *Test Standard for the Measurement of Proton Radiation SEE in Electronic Devices.*	2013
MIL-STD- 750-1	Métodos para realizar el *test* en dispositivos semiconductores que van a trabajar en entornos militares críticos.	2014
MIL-STD-883	Métodos para realizar el test en microcircuitos que van a trabajar en entornos militares críticos.	2014
ESA-ESCC- 25100	*SEE Test Method and Guidelines.* Métodos de test por irradiación con haces de partículas, para equipos electrónicos que van a trabajar en entornos críticos.	2014
ESA-ESCC- 22900	*Total Dose Steady-state Irradiation Test Method.* Métodos de test de Dosis Total para equipos electrónicos que van a trabajar en entornos críticos.	2010
ECSS-Q-ST-60-15C draft	*European Component Space Standardisation* (ECSS) Estándar que especifica los requisitos para componentes espaciales *(*RHA *radiation hardness assurance).* La norma abarca tres principales efectos de la radiación TID, DD y SEE	2012

Tabla 1. Estándares espaciales para los test de irradiación.

El estándar SCC se aplica en las pruebas de protones e iones pesados mientras que el JEDEC sólo sirve para las pruebas con iones pesados. Dado que los SEE afectan a todo tipo de dispositivos y tecnologías, no existe ningún método genérico que se adapte a la perfección a todo tipo de circuitos, sino que en función de las características del entorno de aplicación, se elige el método de irradiación que mejor se adapte a las necesidades.

Todos estos estándares definen que un circuito o sistema digital es tolerante a fallos si puede continuar realizando sus funciones en presencia de errores (debidos a la presencia de fallos transitorios), *hardware* o *software*, es decir, la calidad del servicio que presta dicho circuito o sistema digital no disminuye cuando el sistema se ve afectado por algún error o varios errores.

Es interesante detallar la nomenclatura que utilizaremos en el documento de tesis doctoral para referirnos a los efectos de la radiación ionizante en los sistemas electrónicos digitales.

Un defecto físico o una imperfección en algún componente *hardware* o *software* del circuito se debe principalmente a un cambio físico de las propiedades del componente y puede ocurrir en las distintas etapas del ciclo de diseño. Un fallo en un circuito podría ser una consecuencia de un defecto físico, como por ejemplo un cortocircuito que aparece por un defecto de fabricación en un dispositivo semiconductor. La radiación ionizante, por ejemplo por el paso de protones a través del material semiconductor, puede provocar un pulso de corriente que causaría, a su vez, fallos en el funcionamiento del componente (entre otros, un pulso de corriente transitoria y errónea).

Un error es una consecuencia de un fallo (desde el punto de vista funcional). Así, en el caso del cortocircuito puede ocurrir que una línea del circuito quede fijada a un valor de tensión (adquiriendo un valor lógico permanente). En el caso de radiación ionizante, el pulso de corriente puede provocar un error en el funcionamiento del transistor, generando un valor lógico erróneo y transitorio. A su vez, un error puede provocar un funcionamiento incorrecto del circuito, denominado avería.

El objetivo del diseño de un sistema tolerante a fallos es minimizar el número de averías que se producen cuando existe un fallo y/o detectar y resolver dichas averías.

En este capítulo se analizan las distintas fuentes de fallo en los sistemas electrónicos digitales haciendo hincapié en los fallos permanentes y transitorios. Asimismo se presentan soluciones para resolver los efectos de estos fallos.

2.1. TIPOS Y EFECTOS DE FALLOS

Existen distintas causas que pueden afectar al circuito y producir distintos fallos o errores, a lo largo de su ciclo de diseño y de su tiempo de operación. Los fallos permanentes pueden producirse a lo largo de todo el ciclo de vida del circuito, por errores de diseño, por fallos de fabricación, por fallos debidos al entorno o por envejecimiento del dispositivo. La radiación ionizante acelera el envejecimiento de los componentes electrónicos y puede dar lugar a una degradación en el funcionamiento eléctrico, con implicaciones transitorias o permanentes. Este tipo de daños en un componente determinado puede llegar a inducir un fallo funcional en un equipo e incluso en un sistema completo. Los fallos transitorios no permanecen en el dispositivo, pero pueden provocar averías catastróficas en los sistemas a los que afectan. Principalmente, estos fallos son debidos al entorno hostil en el que operan los circuitos.

Las causas y mecanismos que originan fallos intermitentes son similares a los de los fallos permanentes. De hecho, algunos mecanismos de envejecimiento pueden, en ocasiones, manifestarse como fallos intermitentes hasta que provocan el fallo permanente, de forma que un elevado porcentaje de los fallos intermitentes aparecen antes de los fallos permanentes.

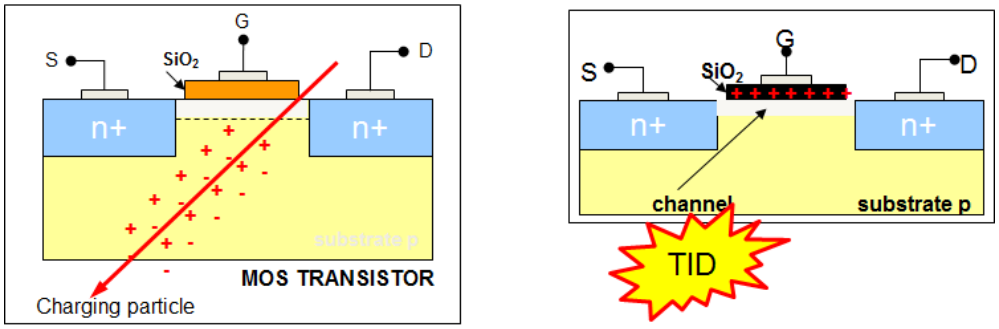


Figura 4. El efecto de las partículas ionizantes [113].

El estudio detallado de posibles efectos de los fallos en el ciclo de operación en condiciones extremas se estudia en este capítulo. En la Figura 4 se muestra el efecto del paso de una partícula ionizante por un dispositivo semiconductor, que implementa un transistor NMOS. La radiación ionizante produce varios efectos:

- Efectos por Dosis de Ionización Total (TID: *Total Ionizing Dose*). Esta Dosis Total Acumulada se debe a las partículas ionizantes y a los fotones que interaccionan con los materiales, y generan pares electrón-hueco en los mismos. Una parte de estos pares se recombinan, pero la otra parte permanece atrapada en forma de cargas eléctricas en las capas del material, Figura 4 b. Este daño es progresivo, llega un momento en que el componente electrónico deja de comportarse adecuadamente, debido a derivaciones paramétricas irreversibles y en ocasiones a un fallo funcional. La velocidad de acumulación de la dosis es normalmente constante, pero la velocidad con que se degrada un componente depende del propio componente. La unidad de medida que se utiliza para definir el nivel de dosis tolerada por cada componente electrónico se expresa en Gray [16],[17]. Se define 1 Gray (Gy) como la cantidad de radiación ionizante necesaria para transferir 1 J de energía a 1 kg del material estudiado. En la definición de esta unidad no es necesario considerar ni la naturaleza de la radiación ni la energía de las partículas o fotones. En cambio, sí es necesario conocer el tipo de material que está siendo irradiado. Por esta causa, se debe hablar de 1Gy (Si), 1Gy (SiO2), etc. Aún sigue siendo ampliamente utilizado el *rad*, que es la centésima parte de 1Gy (1Gy = J/kg = 100 rad).
- *Displacement Damage (DD)* el daño por desplazamiento es otro efecto principalmente causado por las partículas ionizantes en los materiales semiconductores. El primer efecto del DD es la creación de grupos de átomos atrapados en el material, que pueden actuar como generadores, recombinadores o centros de atracción para otros átomos. La degradación lleva a derivas paramétricas irreversibles y eventualmente a fallos funcionales del componente [6][4]. Como en el caso de la TID, las degradaciones deben estudiarse junto a las debidas por otros fenómenos (temperatura, envejecimiento) para garantizar el funcionamiento al final de la vida útil de los componentes. El daño por desplazamiento se mide en RDU (*Radiation Damage Unit*) o unidad de daño por radiación. Se define como la cantidad de energía media que cede un neutrón de 1 MeV al chocar con un átomo de un material determinado. A su vez cada partícula tiene asociado un

NIEL (*Nonionizing Energy Loss*), que define la pérdida de energía no ionizante o deposición de energía en el material asociada a los daños de desplazamiento.

- Efectos transitorios, denominados Single Event Effects, (SEE), generados por la acción de una única partícula. Los efectos de naturaleza transitoria o también llamados efectos puntuales son debidos al paso de una partícula energética por una zona sensible del dispositivo, Figura 4.a. La interacción típica de los iones pesados es la ionización directa de los electrones de valencia, mientras que los protones y los neutrones interaccionan con el núcleo de los atomos semiconductores (mediante choques elásticos e inelásticos) causando la emisión de partículas secundarias, que son las que causan la ionización del material. La consecuencia es un pulso de corriente transitoria, que puede dar lugar a una perturbación funcional del dispositivo (error lógico, transitorio, cortocircuito, etc.) [15][17]. Las partículas responsables de los SEEs son iones procedentes de la radiación cósmica y protones de los cinturones de radiación y de erupción solar. La sensibilidad a los SEEs está aumentando y en la actualidad se consideran un serio problema [4],[5].

Dado que los SEE afectan a todo tipo de dispositivos y tecnologías, no existe ningún modelo genérico de entorno de radiación ionizante que se adapte a la perfección a todo tipo de circuitos y de aplicaciones. Hay muchas condiciones y modos de fallo debidos a SEE, dependiendo de la partícula incidente y del propio componente. Cuando el fallo producido es irreversible, porque la estructura del transistor se ha dañado permanentemente (TID o DD), se denomina "*hard error*" (HE); mientras que los fallos que provocan efectos no permanentes en el circuito se denominan "*soft error*" (SE). Los denominados *soft error* son efectos no destructivos y reversibles para el componente, que pueden afectar a la lógica secuencial (cambio de un bit en una celda de memoria o en un registro) o a la lógica combinacional, y que pueden provocar que un componente interrumpa su funcionamiento normal. Los *hard errors* pueden destruir físicamente al componente, y son efectos funcionales permanentes.

2.1.1. Fallos permanentes

Los fallos permanentes se deben, en su mayor parte, a defectos de fabricación y a los errores del diseñador del circuito, aspectos que llevan asociada una localización espacial. Sin embargo también pueden producirse durante la vida operativa del circuito o sistema, especialmente cuando este comienza a envejecer. Los fallos por fabricación son consecuencia de los defectos puntuales que pueden ocurrir durante el proceso de fabricación de un circuito integrado, en el que puede ser añadido o eliminado una parte del material, causando difusiones erróneas, conexiones incorrectas, propiedades del óxido de silicio inadecuadas, etc. Estos defectos son debidos principalmente a la contaminación con partículas de polvo u otras impurezas de la atmósfera durante los procesos de oxidación, difusión, implantación iónica, etc. que modifican las propiedades de los transistores y por tanto de las puertas lógicas que forman. También la falta o exceso de material pueden causar una capacidad o resistencia no deseada que provocará retardos en las conmutaciones. Estos defectos pueden provocar un fallo de funcionamiento del circuito, que puede modelarse en distintos niveles de abstracción.

Así, en el nivel lógico, los fallos debidos a defectos de fabricación se modelan como un cortocircuito o un circuito abierto en las líneas de conexión de los componentes lógicos. Una vez que aparece un fallo permanente en el sistema, este no se puede eliminar, sólo se puede reemplazar el componente.

Algunos modelos de fallos permanentes son:

- *Stuck-at Fault Model*. El modelo *stuck-at* es uno de los modelos más sencillos. Se asume que el fallo tiene un valor lógico fijo, *stuck-at* 0 o *stuck-at* 1. Estos valores equivalen a un cortocircuito a masa o un cortocircuito a la alimentación respectivamente. Cuando hay un solo fallo en el circuito se utiliza el modelo *single stuck-at* (SSA), en caso de múltiples fallos se utiliza el modelo de *multiple stuck-at*. Este modelo es el generalmente adoptado por todos los fabricantes de circuitos integrados digitales, por su sencillez y por la eficacia de su aplicación a la hora de detectar circuitos con defectos de fabricación.
- *AND/OR Fault Model*. Este modelo se utiliza para describir el comportamiento lógico de dos nodos que están cortocircuitados en el circuito. Con modelo AND se asume que el nodo con el fallo siempre tiene el valor 0, y con el modelo OR se asume que el nodo con el fallo tiene siempre el valor lógico 1.
- *Delay Fault Model*. Este modelo se utiliza para modelar fallos relacionados con retrasos en la propagación de las señales.

Como ya se ha mencionado anteriormente, los requisitos de seguridad (*safety* en la literatura en inglés) de las normas son muy exigentes. Por ejemplo, la norma ISO 26262 establece que para aplicaciones de automoción es necesario que el 90% de los fallos permanentes se detecten incluso cuando el nivel ASIL sea B, que denota un nivel relativamente bajo de seguridad requerido. Obviamente, se requieren porcentajes de fallos detectados más altos cuando el ASIL es C o D.

Los métodos para determinar el nivel ASIL se basan en medidas cualitativas. La norma ISO 26262 tiene tres métricas a tener en cuenta:

1. *Probabilidad de violación de los objetivos de seguridad* - *Probability of violation of safety goals* (PVSG). El objetivo es que esta métrica tenga, para los niveles de sistema, valores menores que 100 FIT (FIT: *"Failure In Time"*). Dependiendo del circuito, los niveles pueden ser más estrictos. Por ejemplo, para un microprocesador que se utiliza en la industria del automóvil, el STM8AF, y que se define en la norma ISO 26262 como *Safety Element out of Context*, el valor absoluto de esta métrica tiene que ser 10 FIT.
2. *Métrica de fallo puntual único* - *Single Point Fault Metric* (SPFM). Esa métrica cuantifica el número de los fallos potencialmente peligrosos detectados y guardados.
3. *Métrica de fallo latente* – *Latent Fault Metric* (LFM). Esta métrica cuantifica el número de los fallos potencialmente peligrosos, que no influyen en el funcionamiento del circuito, los cuales son detectados o no detectados.

En la Tabla 2 se muestra los diferentes umbrales que tienen que cumplir los circuitos para estas métricas de la norma ASIL (ISO-26262).

Métrica	ASIL B	ASIL C	ASIL D
PVSG (1/h)	$< 10^{-7}$ (recomendado)	$< 10^{-7}$	$< 10^{-7}$
SPFM	> 90%	> 97%	> 99%
LFM	> 60%	> 80%	> 90%

Tabla 2. Distintas métricas de fallos permanentes para circuitos en la industria de automoción.

2.1.2. Fallos transitorios

Los fallos transitorios se deben principalmente al efecto de fenómenos externos. Cuando una partícula cargada, como pueden ser las partículas alfa o protones, atraviesa un transistor MOSFET se generan cargas a lo largo de todo el camino que recorre dicha partícula. La carga depositada da lugar a distintos efectos que se puede clasificar en *soft* y *hard-error*.

Los efectos de *soft-error* se describen a continuación:

- *Single Event Upset*, SEU. Este efecto ocurre cuando una partícula impacta en un transistor que pertenece a un biestable o a una celda de memoria y la carga generada es comparable con la carga crítica (carga necesaria para que el transistor conmute), provocando el cambio del valor lógico almacenado. Es un fallo transitorio (reversible cuando se vuelve a escribir el elmento de memoria) que se modela invirtiendo el valor original del bit afectado. El elemento de memoria almacena el valor modificado tras el SEU hasta que se procede a escribir un nuevo valor. Este modelo se denomina *bit-flip* y está ampliamente aceptado por la comunidad científica [8].

- *Single Event Transient*, SET. Un SET se produce cuando una partícula impacta en un transistor perteneciente a lógica combinacional, generando carga que origina un pulso de tensión erróneo de corta duración, del orden de 100 ps. [9].

En tecnologías nanométricas la duración de la transición es comparable al retardo de propagación de la puerta, por lo que el error se puede propagar a través del circuito y llegar a las salidas o almacenarse en un elemento de memoria o varios. De este modo un SET puede dar lugar a uno o a múltiples bit-flip. Estos efectos transitorios están cobrando más interés con el aumento de las frecuencias de funcionamiento en los circuitos actuales y la reducción de los retardos de propagación.

- *Single Event Functional Interrupt*, SEFI. Un SEFI se produce cuando un SE provoca la inversión del valor de un bit en un registro crítico de un sistema de control, produciendo una interrupción del funcionamiento. Por ejemplo, cuando un SE afecta a la memoria de configuración de una FPGA, a la lógica del inicialización asíncrona de un dispositivo, etc. El mecanismo de generación de estos fallos es el mismo que para los SEUs, pero el resultado es distinto porque mientras que un SEU puede no tener un efecto final en la operación del circuito, un SEFI implica por definición un mal funcionamiento que sólo puede corregirse reiniciando la aplicación. En el ejemplo anterior de la FPGA sería necesario reprogramar el dispositivo de nuevo para recuperar su funcionamiento, [10].

- *Multiple Cell Upset/Multiple Bit Upset*, MCU/MBU. Son múltiples fallos transitorios producidos como consecuencia de un único evento. Debido al aumento de la capacidad de integración cada vez es mayor la densidad de transistores en un circuito integrado por lo que está aumentando la probabilidad de que una única partícula genere un fallo transitorio en varios elementos de memoria. Cuando el fallo múltiple se produce en lógica se denomina MBU. Si el fallo se produce en memoria se denomina MBU si afecta a varios bits de una misma palabra, mientras que se conoce como MCU si las celdas afectadas no pertenecen a la misma palabra de memoria, [11].

Los *soft errors* en sí no se consideran definitivamente perjudiciales para los transistores y para la funcionalidad de los circuitos a diferencia del *hard errors*.

- *Single Event Latch-up*, SEL. Un SEL sucede cuando, en tecnología CMOS con substrato común, la carga generada por la partícula ionizante activa transistores bipolares parásitos, que se forman entre el sustrato y las diferentes zonas dopadas de los transistores, abriendo un camino entre la alimentación y la masa y provocando un cortocircuito. Para eliminar este fallo es necesario interrumpir la alimentación del circuito. Este efecto es muy peligroso, puesto que las grandes corrientes que se generan pueden dañar los componentes del circuito permanentemente y dejarlo inservible. El SEL puede evitarse utilizando tecnología SOI (*Silicon on Insulator*) puesto que así se eliminan los transistores parásitos o aplicando técnicas de *layout* para aumentar la resistencia de diseños CMOS a este efecto. De hecho, en las aplicaciones espaciales críticas se utilizan generalmente tecnologías resistentes a SEL, [9],[20].

- *Single Event Gate Rupture*, SEGR se produce cuando una partícula cargada atraviesa la puerta de un transistor MOS creando un plasma conductor que cortocircuita los terminales de puerta y el canal [19].

- *Single Event Burn-Out*, SEBO se produce cuando un transistor de potencia (bipolar o de efecto campo) se encuentra en situación de corte y una partícula cargada viajera crea un plasma conductor entre los extremos (drenador y fuente, colector y emisor). El cortocircuito que se produce conlleva a la destrucción por temperatura del dispositivo [18].

De todos los efectos descritos los SEUs son los efectos más notables y más frecuentes debido a las características de las tecnologías actuales. Sin embargo, con el incremento en las frecuencias de funcionamiento se prevé una tasa creciente de SETs por lo que estos efectos se están convirtiendo en un problema cada vez más importante. Recientemente, se han realizado investigaciones en las que se estudian los efectos de los SETs como si fuesen equivalentes a múltiples *bit-flips*.

Hay que ser conscientes del coste que suponen los *soft errors*, que incluye el tiempo perdido en su análisis y diagnóstico, la reducción de la fiabilidad y la disponibilidad del sistema y, obviamente la reducción de la satisfacción del cliente. Los fallos transitorios en los sistemas digitales tienen una diferencia respecto a los fallos transitorios en los componentes digitales. Cuanto más grande y complejo es el sistema, la tolerancia a fallos ya no es sólo función de la tolerancia de sus componentes individuales. Es un problema más complejo que debe ser analizado en detalle para proponer soluciones eficientes y económicas.

El *test de operación (test on-line)* se centra en estudiar los fallos originados durante la fase de operación de un circuito, su detección y las medidas de respuesta que aplica el sistema. Como consecuencia del desarrollo de la tecnología, los circuitos digitales son cada vez más sensibles a las condiciones externas, aumentando el número de fallos transitorios que afectan a un circuito digital durante su fase de operación. Es por ello fundamental que los sistemas y circuitos que van a operar en entornos críticos se diseñen, desde el principio, con una alta tolerancia a fallos.

2.2. TOLERANCIA A FALLOS EN CIRCUITOS Y SISTEMAS DIGITALES

2.2.1. Métricas de confiabilidad

El proceso de diseño de un sistema tolerante a fallos consta fundamentalmente de dos fases, la aplicación de técnicas de mitigación de los efectos de los fallos y, de forma

iterativa, la evaluación del nivel de robustez alcanzado por todo el sistema digital, hasta que cumpla con las especificaciones de confiabilidad.

Esta propiedad, la calidad del servicio, se denomina confiabilidad y es el objetivo que se persigue durante el diseño de un sistema tolerante a fallos. Para evaluar la confiabilidad del sistema tolerante a fallos, se estudian las siguientes características:

• Fiabilidad: es la probabilidad de que un sistema esté trabajando en forma continua (sin interrupciones) en un intervalo de tiempo o a lo largo de toda su vida útil. Se mide como la probabilidad de que el sistema funcione correctamente a lo largo de un intervalo de tiempo (t_0, t) supuesto que funcionaba en el instante t_0. Cuando un sistema es tolerante a fallos puede continuar con su funcionamiento aunque ocurra un fallo, por lo que tendrá una alta fiabilidad.

Supongamos que tenemos N componentes idénticos que funcionan correctamente en el instante t_0, sea $N_0(t)$ el número de componentes que siguen funcionando correctamente en el instante t, y $N_f(t)$ el resto, es decir los que han fallado a lo largo del intervalo (t_0-t). La fiabilidad de los componentes en el instante t vendrá dada por:

$$R(t)= N0(t)/N=N0(t)/(N0(t)+Nf(t))$$

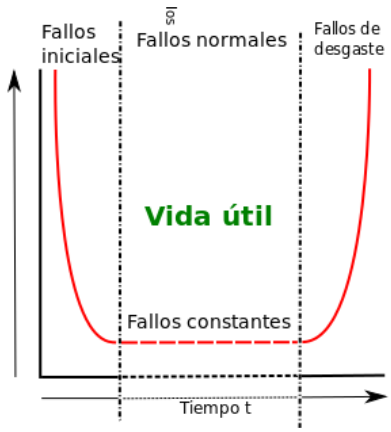
Análogamente se puede definir la probabilidad de que un componente falle (Q(t))

$$Q(t)= Nf(t)/N=Nf(t)/(N0(t)+Nf(t))$$

Por lo que deberíamos definir la fiabilidad como:

$$R(t)= 1-Nf(t)/N$$

Derivando respecto al tiempo se obtiene la función de riesgo, o tasa de averías, uno de los parámetros para medir la calidad de los sistemas en lo referente a la tolerancia a fallos. La función de riesgo, dependiendo de la naturaleza del sistema, tiene diferentes formas. Para componentes electrónicos tiene la forma de curva de bañera y en esta destacan tres zonas diferentes.



En la zona de mortalidad inicial fallan los componentes inicialmente defectuosos (mortalidad infantil). La zona de estabilidad o de vida útil es la zona de mayor duración donde la tasa de fallos permanece sensiblemente constante. La zona de desgaste es la zona donde a partir de cierto momento los componentes han finalizado su vida útil y empiezan a fallar por desgaste [13].

Sin embargo, una alta fiabilidad no implica necesariamente que el sistema sea tolerante a fallos. Por ejemplo, en un sistema no tolerante, pero con una tasa de averías muy baja el valor de la fiabilidad sería alto. O al revés, con una alta fiabilidad el sistema puede ser no tolerante a algún tipo de fallos concreto.

El valor constante de la tasa de averías correspondiente a la vida útil del dispositivo se expresa como el número de averías ocurrido en 10^9 horas (FIT).

$$1FIT = \frac{1\text{Averia}}{10^9\text{horas}}$$

Este valor depende de factores como la madurez del proceso de fabricación utilizado, el entorno de aplicación, la complejidad del circuito, la cantidad de *tests* realizados después de la fabricación, el efecto de la temperatura, etc. Típicamente, en los circuitos actuales, sin mecanismos de tolerancia a fallos, el valor de la tasa de averías está comprendido entre 1000 y 50.000 FIT por circuito integrado.

• Disponibilidad: se mide como la probabilidad de que el circuito opere correctamente y esté disponible para ejecutar sus funciones en un instante de tiempo dado. Esta propiedad, a diferencia de la fiabilidad, depende de la velocidad a la que los fallos pueden ser reparados. La disponibilidad tiene tres características:

- El tiempo medio para que se produzca una avería (MTTF, *Mean Time To Failure*). Se define como el tiempo que se espera que el sistema funcione correctamente antes de que ocurra la primera avería y se puede calcular como

$$\int_0^\infty R(t)dt$$

Donde R(t) es la fiabilidad del sistema, la probabilidad de que funcione correctamente durante un intervalo de tiempo determinado.

- El tiempo medio entre averías (MTBF, *Mean Time Between Failure*). Es el tiempo medio transcurrido entre dos funcionamientos erróneos del sistema. Su valor suele ser del orden del MTTF y se puede estimar como la relación entre un cierto intervalo de tiempo y el número de fallos ocurridos en ese período.

- El tiempo medio para reparar el sistema (MTTR, *Mean Time To Repair*). Es el tiempo medio necesario para reparar el sistema de una avería.

Asumiendo que una vez reparado el sistema éste recupera totalmente sus capacidades, se puede establecer que

$$MTTR = MTBF - MTTF$$

El tiempo medio entre fallos es la suma del tiempo necesario para reparar el sistema más el tiempo esperado para que suceda otro error. Por ejemplo, en un sistema que se recupera rápidamente de un error pero en el que la tasa de fallos es alta, la fiabilidad sería baja mientras que la probabilidad de que en un instante determinado el sistema funcionase correctamente sería alta.

• Sensibilidad: es la capacidad de que un tipo de radiación y energía producida en el material provoque un fallo o una avería. La sensibilidad de un dispositivo frente a SEEs, provocados por partículas ionizantes, depende de su LET (*Linear Energy Transfer Threshold*) que es la energía de ionización que pierde la partícula por unidad de longitud recorrida. Esta magnitud mide la energía que deposita una partícula por unidad de masa y volumen del material de impacto, y por distancia recorrida en su interior. Las unidades habituales de LET asociada a una partícula son $MeV \cdot cm^{-2} \cdot mg^{-1}$. Debido a que diferentes tipos de radiación tienen diferentes características de interacción con la materia, la sensibilidad de detección es distinta. Las partículas alfa, protones y otras partículas pesadas provocan una radiación con alto valor de LET, mientras que los electrones se consideran radiación de bajo valor de LET. Para caracterizar correctamente la sensibilidad de los componentes frente a iones pesados se relaciona la sección eficaz con el LET de los iones incidentes, resultando la llamada curva de "*Weibull*". Esta se obtiene experimentalmente irradiando con varios tipos de iones, variando sus energías y variando los ángulos de incidencia. Así, la sensibilidad del dispositivo a un tipo particular de radiación puede ser caracterizada en términos de sección eficaz. La Figura 5 muestra la sensibilidad del dispositivo a la radiación, en este caso procedente de iones pesados, en función del LET del ion incidente. De esta manera

se expresa la sección eficaz como la probabilidad de que ocurra un fenómeno transitorio o SEE por partícula incidente y por superficie, en función del LET asociado.

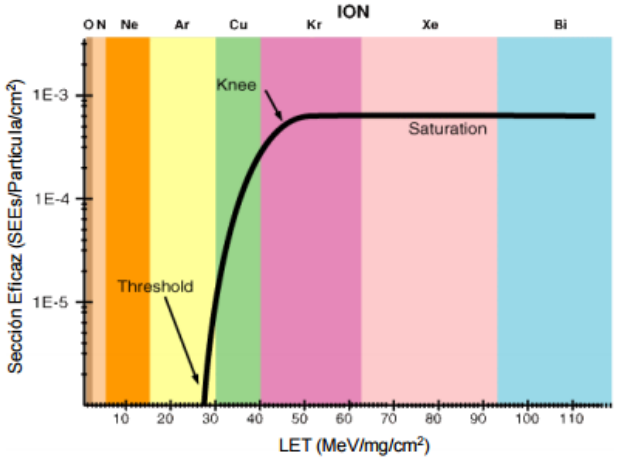


Figura 5. Curva de sensibilidad LET/Sección eficaz [17].

La Figura 5 muestra los valores de la sección eficaz en función del LET. Dicha sección eficaz es el número de SEEs, observados en el material analizado, por partícula por centímetro cuadrado. En diferentes colores está marcado el LET asociado a cada tipo de ion [17]. Hay que destacar que para un valor menor del LET el dispositivo acumula más TID [14] aunque presenta menor sección eficaz.

• Seguridad (*safety*): es una medida de la capacidad de un circuito de no dañar a otros circuitos ni a su entorno cuando se produce un mal funcionamiento. Por ejemplo, un fallo en el sistema de ABS de los automóviles no evita que se pueda seguir frenando. La seguridad se mide como la probabilidad de que un circuito funcione correctamente o de que en caso de que falle lo haga de un modo seguro.

• Capacidad de ejecución (*performability*): es la probabilidad de que el sistema funcione con cierto nivel de calidad en un instante de tiempo determinado, es decir, de que al menos algunas funciones del sistema permanezcan activas. Este atributo se aplica en los sistemas en los que un fallo no interrumpe el funcionamiento pero disminuye su calidad. Por ejemplo, en un sistema multiprocesador en el que uno de los procesadores falle, el sistema puede continuar su ejecución aunque pierda prestaciones como velocidad o memoria disponible.

• Mantenibilidad (*maintainability*): mide la facilidad con la que un circuito se puede reparar una vez que ha fallado. La mantenibilidad se cuantifica como la probabilidad de que un sistema con avería pueda estar de nuevo operativo en un instante de tiempo determinado. El proceso de restaurar un sistema incluye la localización del problema, la reparación física y la puesta en funcionamiento. La tarea de localizar el error suele ser el factor limitante. Como se verá más adelante, algunas de las técnicas de tolerancia a fallos existentes proporcionan métodos de diagnóstico automático de errores, de forma que la tolerancia a fallos puede mejorar considerablemente la mantenibilidad de un sistema.

• Testabilidad (*testability*), Diseño para Testabilidad (DFT) es la capacidad que tiene el sistema para comprobar partes del circuito y la calidad de sus funciones. La testabilidad permite medir la facilidad con la que se puede realizar un test sobre el diseño. Esta característica permite localizar y detectar fallos lo que repercute en la mantenibilidad del sistema mejorándola.

La cobertura de recuperación de los fallos es la probabilidad de que el sistema se recupere tras un fallo. Esta función de probabilidad depende del instante de tiempo en el que se produce el fallo, su localización, el tipo del mismo y la actividad del sistema. El método más común para calcular la cobertura de fallos consiste en desarrollar una lista

de todos los posibles fallos que pueden afectar al sistema y analizar cuántos de esos fallos se pueden detectar, localizar o recuperar. Entonces, la cobertura de fallos sería la relación entre los fallos detectados o recuperados y el número total de fallos. También suele denominarse tasa de error.

$$Cobertura\ de\ fallos = \frac{Número\ de\ Averías}{Número\ de\ fallos}$$

En resumen, los objetivos que se persiguen al diseñar un circuito son, además de que ejecute correctamente su función y que sea efectivo en coste, que sea fiable, fácil de testar, fácil de recuperar y seguro, es decir, que sea confiable y por lo tanto tolerante a fallos.

2.2.2. Soluciones para resolver el efecto de los fallos en los circuitos digitales

Las técnicas de tolerancia a fallos pueden consistir en soluciones tecnológicas o en técnicas basadas en el diseño, pero siempre se basa en algún tipo de redundancia.

Las soluciones tecnológicas consisten en la modificación del proceso de fabricación con el objetivo de que el circuito fabricado sea menos sensible a los efectos provocados por la radiación, disminuyéndose la tasa de fallos e incluso eliminándose por completo algunos de los efectos. Existen tecnologías orientadas a ser robustas frente a las radiaciones, que se utilizan en las partes críticas de las aplicaciones aeroespaciales. Estas tecnologías, denominadas *radiation-hard* o *rad-hard*, garantizan la tolerancia a los fallos por dosis total y una menor sensibilidad a los fallos SEL que son los que pueden dar lugar a fallos permanentes o a la destrucción del circuito, a la vez que proporcionan una mayor resistencia a los fallos SEUs. La tecnología *rad-hard* requiere un proceso de fabricación especial mucho más caro que el convencional, por lo que se usa únicamente en aplicaciones muy específicas.

Como ejemplo, para los satélites de baja inclinación (# 28 grados) *Low Earth Orbit* (LEO) <500 km, en ambos hemisferios norte y sur, las tasas típicas de dosis debidas a los electrones y los protones atrapados por los cinturones de Van Allen son 100-1000 rad (Si) al año. Para los satélites en inclinaciones superiores a LEO, en ambos hemisferios norte y sur, las tasas de dosis típicas son 1000-10.000 rads (Si) al año debido a la mayor cantidad de electrones atrapados.

Hay tres categorías de componentes electrónicos según su grado de robustez, en la Tabla 3 se indica sus tolerancias máximas, así como el grado de calificación que proporciona el fabricante.

Componente	***Nivel de endurecimiento del diseño***	***Niveles de robustez***			***Test de evaluación***
		Dosis total (krad)	***Umbral LET para SEUs MeV/mg/cm²***	***SEU Error Rate bit/día (típico)***	
Comercial	Limitado	de 2 a 10	5	10^{-5}	Por cliente
Rad tolerante	dureza rad hasta un cierto nivel	20 - 50	20	10^{-7}–10^{-8}	Por cliente
Rad-hard	especial nivel de dureza	200krad - >1Mrad	80-150	10^{-10} a 10^{-12}	Obleas testeadas

Tabla 3. Distintos tipos de componentes y sus características de robustez

El uso de los componentes *rad-soft* no reduce significativamente el coste, pero aumenta mucho el riesgo. Como se puede deducir, no hay componentes que sean ideales

para todas las aplicaciones. Los componentes comerciales son útiles sólo para aplicaciones comerciales de bajo coste, donde la última tecnología y la alta velocidad tienen prioridad sobre el entorno crítico del componente. El blindaje de estos dispositivos en aplicaciones espaciales, especialmente para los efectos de SEUs y SELs resulta ser un esfuerzo casi siempre inútil.

Las técnicas de tolerancia a fallos por diseño permiten utilizar las tecnologías CMOS comerciales alcanzando unos niveles de tolerancia a fallos aceptables. Estas técnicas se pueden clasificar en función de la clase de efecto frente al que se quiere aumentar la fiabilidad. Para los fallos del tipo TID se aplican soluciones a nivel físico, modificando el *layout* del circuito.

Las soluciones para corregir los efectos de fallos SEUs consisten en introducir redundancia en el circuito integrado. Esto puede hacerse a nivel de transistor modificando la celda de memoria, o a nivel más alto, RT (*Register Transfer*) o de sistema, introduciendo estructuras redundantes durante el diseño del circuito [21][22][23].

• Redundancia hardware es una técnica de endurecimiento aplicada para mitigar el efecto de los fallos transitorios. La replicación física del componente es seguramente la redundancia más frecuente en los sistemas digitales. Existen varias formas básicas de redundancia de hardware.

- Redundancia pasiva: los componentes redundantes se utilizan dentro del sistema para enmascarar los efectos de los componentes con defectos corrigiéndolos, pero no informando al exterior de su presencia. La gran mayoría de estas técnicas utilizan el mecanismo de votación por mayoría. Este mecanismo consiste en replicar el componente hardware en cuestión y elegir como salida correcta del circuito aquella que se obtenga mayoritariamente. Así, si uno de los módulos replicados tiene un fallo y la salida del resto de módulos es correcta el fallo se enmascara, y generalmente se corrige.

  - TMR (*Triple Modular Redundancy*) la técnica con N = 3 es la más habitual. Las ventajas de este método son la recuperación automática de un circuito afectado por un SEU, una alta tolerancia a los SEUs y que la conversión de un sistema no redundante a uno redundante es muy sencilla. Esta técnica es capaz de mitigar los efectos de fallos simples.

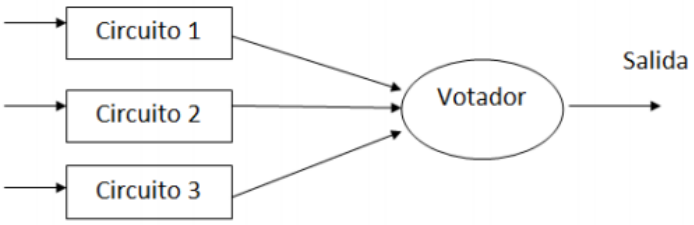


Figura 6. El circuito TMR típico.

El problema de esta técnica es que, aunque realiza la corrección de los fallos de tipo SEU, en caso de que exista un fallo tipo SET no es capaz de mitigar su efecto, que se propaga por todas las réplicas. Además, esta técnica es muy costosa, tanto en términos del área como de energía. El votador para esta técnica es un circuito combinacional, y si el TMR corrige los SEUs, el votador no debería fallar. Si se quieren corregir fallos tipo SET, habría que modificar la sincronización de las réplicas, así como endurecer también el votador, Figura 7.

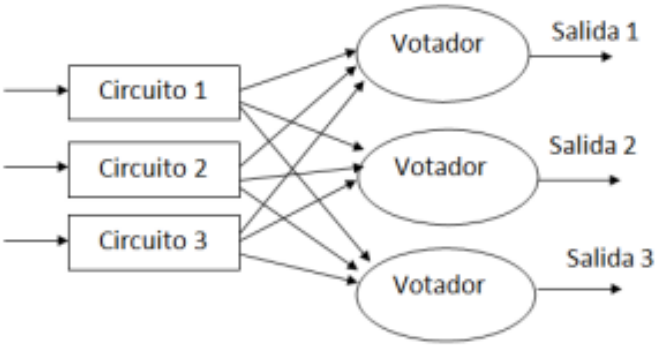


Figura 7. El circuito TMR con la redundancia en el detector.

- NMR (*N-Modular Redundancy*). La forma más común de redundancia hardware pasiva consiste en replicar N veces el bloque que se quiere hacer tolerante a fallos, siendo N un número impar, y realizar una votación por mayoría de las salidas para obtener la salida correcta del módulo. Esta técnica es capaz de mitigar el efecto de fallos múltiples, $\log_2 n$. Sin embargo, la desventaja de la redundancia con votación sigue siendo la misma: esta técnica, en cualquiera de sus variantes, genera un único resultado de la votación sin redundancia, lo que la hace sensible a fallos tipo SET.
- TMR selectivo. Es una de las técnicas en el cual se triplica sólo una parte del circuito. Mediante esta técnica, el diseñador debe identificar qué regiones del diseño es más conveniente proteger haciendo un estudio previo de las regiones que son más o menos críticas en el sistema, para así protegerlas en primer lugar [23]. Con esta técnica se reduce el área utilizada y de esta forma se ahorran los costes de la protección. En cuanto a las desventajas, esta técnica puede introducir retardos adicionales y, además, quedan desprotegidas las señales de reloj y de inicialización asíncrona, así como la lógica interna del dispositivo afectada por los SEFIs [25].

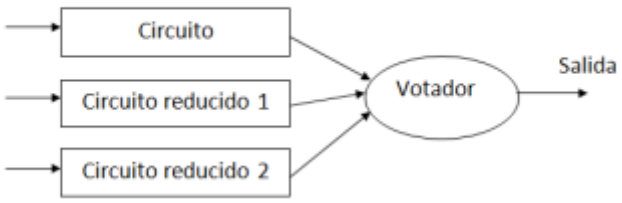


Figura 8. El circuito TMR selectivo.

- BTMR *(Block Triple Modular Redundancy)* y DTMR (*Distributed Triple Modular Redundancy*). La técnica BTMR necesita realimentación para corregir los errores y, por lo general, no se puede aplicar la corrección interna desde las salidas del votador, así que los errores no se

corrigen y por tanto no es una técnica efectiva. La técnica DTMR, es igual que el TMR pero sin replicación de la señal de reloj [25].

En general, estas técnicas de redundancia pasiva son más efectivas cuanto mayor es el grado de atomización que se emplea. Es mucho más efectivo endurecer mediante TMR (o NMR) los biestables más críticos del circuito, y corregir su efecto de forma local para evitar su propagación innecesaria.

- Redundancia activa. Se utiliza para la detección *on-line* de los errores del sistema que permiten la aplicación de alguna acción correctora externa, como por ejemplo sustituir el componente erróneo por otro alternativo. Esta técnica requiere una reconfiguración del sistema para continuar funcionando correctamente. Estas técnicas se usan en aplicaciones en las que se permite un funcionamiento erróneo durante un tiempo limitado (mientras se reconfigura el circuito). La mayor parte de estas técnicas constan de detectores de error y módulos de repuesto. Los módulos de repuesto permanecen inactivos hasta que se detecta un error en un componente hardware y éste se sustituye por su módulo de repuesto.

A continuación se enumeran varios métodos de redundancia activa.

- Duplicación con comparación (*Duplication with Comparison*). Un método de redundancia activa, que es capaz de detectar un fallo pero no de localizarlo. Los fallos ocurridos en el comparador [22] pueden ser detectados mediante la utilización de códigos Dual Rail en bloques totalmente auto-comprobables (*Totally Self-Checking)[23]*.

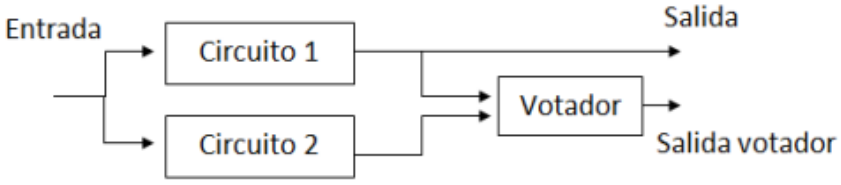


Figura 9. El circuito duplicación con comparación (DWC).

- Repuestos en espera (*stand-by sparing*), técnica que se muestra en la Figura 10. Esta técnica consiste en tener el módulo crítico replicado, sin embargo, sólo uno está en funcionamiento en cada instante, el resto están inactivos en espera. Cuando un módulo falla, se pone en marcha el de repuesto. Los detectores de error seleccionan la entrada activa del multiplexor. Una ventaja de este método es que un sistema con N módulos idénticos puede proporcionar mucha tolerancia a fallos con un número de repuestos bastante inferior a N, ya que no es necesario tener un repuesto por módulo. Por otra parte, tiene las desventajas de que el multiplexor no es tolerante a fallos y que es un sistema bastante complejo y costoso [25].

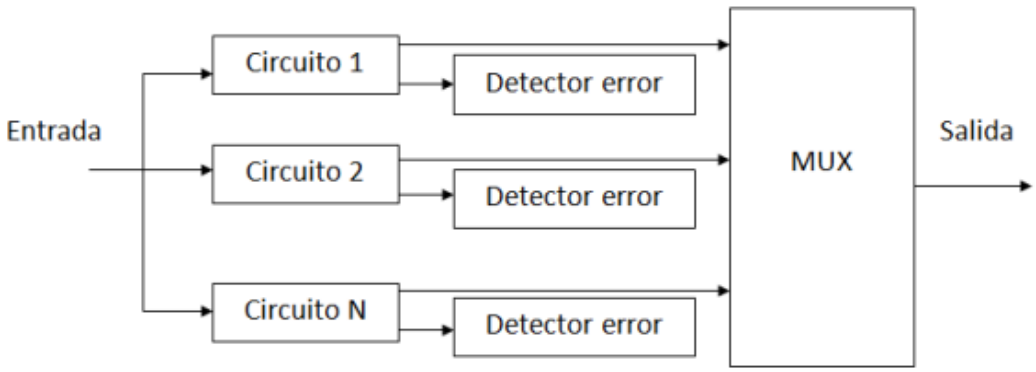


Figura 10. El circuito de repuestos en espera.

• Métodos híbridos. En ellos se realiza el enmascaramiento de fallos, pero, el sistema también puede reconfigurarse en caso del fallo. Las diferentes técnicas de estos métodos son la redundancia n-modular con repuestos, la redundancia con autoeliminación, o la arquitectura triple-dúplex [25].

• Redundancia Temporal es una técnica eficaz para detectar o enmascarar fallos transitorios basándose en el carácter temporal de estos. Se realiza mediante el uso repetido de un componente en presencia de fallos. Además, las técnicas de redundancia temporal requieren generalmente pocos elementos hardware extra para su implementación y control a diferencia de las soluciones con redundancia hardware y de información. Por lo tanto, esta técnica es adecuada en circuitos en los que el hardware es un recurso más crítico que el tiempo, ya sea por el coste, el tamaño o el peso. La redundancia temporal puede implementarse de dos formas distintas:

- Repitiendo los cálculos en varios instantes de tiempo para luego comparar los resultados obtenidos.
- Almacenando los datos en distintos instantes de tiempo sin necesidad de repetir el cálculo.

Esta técnica necesita un tiempo adicional, por lo que se suele utilizar en circuitos cuya ejecución no sea larga, pero puede ser utilizada también para detectar no sólo errores transitorios sino errores permanentes [22].

• Redundancia de información. Este tipo de redundancia consiste en añadir información adicional a los datos para permitir la detección o corrección de errores. A continuación se citan algunas de estas técnicas.

- Códigos de paridad. La técnica del bit de paridad consiste en guardar un bit adicional por cada byte de datos, y en la lectura se comprueba si el número de unos es par (paridad par) o impar (paridad impar), detectándose así el error.
- Corrección de errores (*Error Correcting Codes*, ECC). La técnica ECC, que permite detectar errores de 1 a 4 bits y corregir errores que afecten a un sólo bit, se usa sólo en sistemas que requieren alta fiabilidad.
- Detección de errores *(Error Detection Codes* EDC).
- Checksum.
- Códigos de Berger.

Cada técnica de redundancia de información está caracterizada por su distancia mínima de Hamming, que indica el número de fallos que dicha técnica es capaz de detectar. La distancia de Hamming entre dos datos indica el número de bits en los que

ambos datos difieren, de forma que no existen dos datos válidos que difieren en un número de bits menor que la distancia mínima de Hamming.

Los códigos de corrección de errores son el mecanismo más habitual para reducir la tasa de errores en memorias, en particular en SRAM al ser este tipo de memoria el más sensible a SEUs. Sin embargo, hay que tener en cuenta, que el área necesaria es mayor para los códigos correctores que para los que únicamente detectan errores, en especial si corrigen errores múltiples, por lo que el diseñador debe encontrar un compromiso entre la penalización en el área, la complejidad del sistema y la tolerancia a fallos que se obtiene [26].

• Redundancia software

En aplicaciones que usan microprocesadores, se puede utilizar la redundancia software. Esta técnica proporciona flexibilidad y no requiere ninguna modificación hardware, por lo que se puede usar para detectar y corregir fallos permanentes y transitorios con un coste muy bajo de implementación. Este mecanismo consiste en replicar parte del código o añadir código extra que realice funciones de comprobación, de modo que implica un incremento en los requisitos de memoria. Puede basarse en:

- Pruebas de consistencia de alguna magnitud (por ejemplo el rendimiento) mediante software.
- Pruebas de hardware. Habilitar procesos que se dediquen a verificar el hardware.
- Varias versiones de programación. Consiste en diseñar y codificar de manera independiente el módulo software N veces y aplicar los métodos como la redundancia hardware pasiva explicada anteriormente.

Los fallos pueden generar errores en los datos o en el flujo de control de la ejecución y existen distintas técnicas orientadas a tratar cada uno de estos problemas [27][28]. Normalmente, se aplican varias técnicas conjuntamente para cubrir fallos tanto en los datos como en el control de la ejecución.

• Test funcional (fallos permanentes)

Realización de un banco de pruebas (testbench) completo que cubra todos los posibles casos de funcionamiento del sistema. Realizar la optimización y depuración del test para la reducción de falsos errores [29]. Cuando se evalúan los sistemas tolerantes a fallos, la cobertura de fallos y errores es muy importante. La evaluación de esta cobertura también puede realizarse por el modelado o por el test, llamado en este caso inyección de fallos. La cobertura de un test funcional es la fracción de los fallos que son puestos en evidencia por el test.

• Técnicas específicas para FPGAs/SoC

Un SoC es un sistema que integra distintos componentes como un procesador empotrado, lógica de usuario y memoria. Frecuentemente, el diseño de estos sistemas se basa en la reutilización de componentes ya existentes, por lo que no es posible modificar notablemente dichos componentes con la inserción de técnicas de redundancia para conseguir que sean tolerantes a fallos. La solución que se plantea en la literatura consiste en aplicar técnicas de endurecimiento de forma global a la arquitectura del sistema. Una de las técnicas propuesta es la detección de fallos orientada al endurecimiento de SoCs. La solución consiste en añadir un bloque

adicional encargado de observar las operaciones de otros componentes y combinarlo con técnicas de redundancia software para el control de flujo y datos (no se modifica la lógica de cada componente individualmente). Esta técnica puede aplicarse también en el endurecimiento de SoPCs, implementando el módulo de detección en la lógica programable disponible en el sistema [43].

También los dispositivos lógicos programables, como las FPGAs, requieren de técnicas de tolerancia a fallos específicas. El estudio de los efectos de SEEs en FPGAs y el desarrollo de técnicas de mitigación y evaluación de fallos para dichos dispositivos es un campo de investigación muy amplio y todavía abierto, que queda fuera del alcance de esta tesis. Sin embargo, con el objetivo de proporcionar una visión global de la temática relacionada con la tolerancia a fallos, se presenta una breve revisión de los problemas que plantean estos dispositivos con respecto a los fallos debidos a la radiación y se muestran las posibles soluciones.

Las FPGAs proporcionan una solución interesante en numerosas aplicaciones, por sus prestaciones y coste. En especial, las FPGAs basadas en SRAM, son muy sensibles a SEEs, pero resultan ser dispositivos muy potentes que se usan cada vez más en numerosas aplicaciones, gracias a que pueden ser reprogramadas tantas veces como sea necesario modificando la memoria de configuración (que es una memoria SRAM). Un SEU en la memoria de configuración puede modificar las funciones lógicas implementadas en el dispositivo o las interconexiones (produciendo cortocircuitos, circuitos abiertos o asignaciones erróneas de señales), interrumpiendo el funcionamiento normal y provocando, por tanto, SEFIs. Los bits de la memoria de configuración suponen generalmente más de un 95% del número total de bits susceptibles de sufrir un SEU en una FPGA, por lo que, es evidente la necesidad de prestar una especial atención a este tipo de errores.

La tolerancia a fallos en FPGAs se puede lograr de dos formas diferentes:

• Aplicando soluciones relacionadas con la tecnología y la arquitectura interna, es decir, desarrollando FPGAs fabricadas con elementos tolerantes a fallos. Esta solución conlleva la fabricación de un nuevo chip endurecido (*rad-hard* FPGAs) por lo que es muy costoso. Además las FPGAs *rad-hard* ofrecen menos prestaciones que las comerciales. Las tecnologías con las que se fabrican proporcionan menor densidad de integración y no son reconfigurables si se eliminan completamente los efectos SEUs de la memoria de configuración (FPGAs de antifusibles). Por lo tanto, su uso se limita a aplicaciones de seguridad crítica con alto presupuesto (por ejemplo, en sistemas espaciales).

• Aplicando técnicas de endurecimiento al diseño a alto nivel, introduciendo redundancia en el circuito implementado, en el proceso de rutado, redundancia temporal, etc. Estas técnicas permiten utilizar FPGAs comerciales con altas prestaciones y un bajo coste. Las técnicas que pueden aplicarse para endurecer la lógica de usuario, como los registros, las memorias empotradas, multiplexores, etc., son las mismas técnicas que se pueden aplicar en ASICs [44].

Sin embargo, para resolver el problema de los fallos en la memoria de configuración es necesario desarrollar nuevas soluciones, como la reconfiguración periódica del dispositivo o, en inglés, scrubbing, tanto completa como parcial, o las técnicas de tolerancia a fallos en el rutado [45].

Recientemente, han surgido las FPGAs basadas en memoria tipo Flash, que utilizan transistores MOSFET de puerta flotante como elemento de configuración. Estas FPGAs presentan una menor sensibilidad a la radiación ionizante que las FPGAs basadas en memoria SRAM [44], además de que los fabricantes están incluyendo dispositivos de capacidades medias-altas y tecnologías de ultra-bajo consumo. Aunque

no están destinadas específicamente a aplicaciones aeroespaciales, es cierto que se están incorporando a estos sistemas por sus ventajas.

2.3. MÉTODOS PARA EVALUACIÓN DE LA TOLERANCIA A FALLOS MEDIANTE INYECCIÓN DE FALLOS

Como se ha dicho anteriormente, el endurecimiento de sistemas digitales frente a la radiación ionizante implica dos etapas. La primera, descrita en el apartado 2.2, es la inserción de estructuras de mitigación de fallos transitorios (debidos a dicha radiación). La segunda etapa, iterativa, es la evaluación de la tolerancia alcanzada en la etapa primera mediante algún método de comparación entre un circuito libre de fallo y uno con fallos. La generación de réplicas del circuito con fallos insertados puede acometerse en distintos niveles de abstracción. Dependiendo del circuito o componente que se está validando, se pueden inyectar los fallos sobre una descripción del circuito y sobre un componente comercial.

Si se dispone de una descripción del circuito (sea en un lenguaje de descripción de *hardware*, alto nivel de abstracción, o sea una lista de puertas lógicas, niveles más bajos de abstracción) es posible realizar la inyección de fallos en los elementos más sencillos del circuito y se puede realizar mediante emulación *hardware* y mediante simulación.

La inyección de fallos sobre un componente comercial se aplica para analizar la confiabilidad de dispositivos finales, calificación, para medir el comportamiento de las técnicas de mitigación de fallos frente a fallos reales o cuando no se dispone de una descripción del circuito. Las técnicas más comunes son la inyección de fallos físicos con un acelerador de partículas o con un láser e inyección de fallos lógicos. Estas técnicas se aplican sobre el circuito mientras esté funciona en modo normal, por lo que son muy apropiadas para evaluar aplicaciones en tiempo real.

A continuación se describen los diferentes métodos de inyección de fallos.

2.3.1. Inyección de fallos por simulación

Esta técnica se puede realizar en la etapa de diseño del circuito. Se utilizan modelos en VHDL, Verilog u otros lenguajes que pueden ser desarrollados especialmente para la campaña de inyección de fallos. La inyección de fallos mediante simulación en descripciones del circuito puede implementarse modificando la descripción VHDL del circuito, utilizando comandos de un simulador VHDL comercial o modificando las herramientas de simulación. Una de las desventajas más grandes de la inyección de fallos por simulación es el tiempo necesario para realizar la campaña. Una simulación completa es muy larga para cualquier circuito de tamaño razonable y el desarrollo de los modelos a menudo requiere un esfuerzo considerable.

Existen varias formas para inyectar fallos mediante simulación del sistema bajo prueba.

1. Modificar el valor lógico de un elemento interno del circuito mediante comandos del simulador utilizado. Esta técnica tiene la ventaja de ser poco intrusiva al no modificar el código del modelo del sistema.
2. Modificar el código del modelo para inyectar fallos y observar el comportamiento del mismo. Añadir los componentes a la descripción del circuito que permitan modificar y observar el valor lógico de un elemento interno del circuito.
3. Introducir cambios en el código fuente de la descripción hardware del circuito. La variedad de cambios que se pueden introducir es amplia: reemplazar un componente en una descripción estructural (por ejemplo

cambiar un *AND* por un *OR*) o modificar estructuras de control en una descripción estructural (por ejemplo reemplazar una condición por un valor fijo *TRUE* o *FALSE*, perturbar el valor de una señal, etc.).

4. En el caso de modelos VHDL una técnica poco invasiva consiste en inyectar los fallos modificando la definición de los tipos de datos y las funciones.

Existen varias herramientas para modificar automáticamente el código de descripción hardware y realizar campañas de inyección de fallos basadas en simulación [31]- [38].

2.3.2. Inyección de fallos por emulación

Las técnicas para la evaluación de la tolerancia a fallos de un circuito basadas en emulación consisten en realizar la inyección de fallos sobre un prototipo del circuito que se implementa en un dispositivo lógico reprogramable. La emulación proporciona información sobre el funcionamiento del circuito en presencia de fallos y permite evaluar, durante el ciclo de diseño, el comportamiento de los mecanismos de tolerancia a fallos integrados en el circuito. Sin embargo, en las técnicas basadas en emulación, los fallos se insertan sobre una plataforma hardware de emulación, una FPGA, lo que permite realizar la campaña de inyección a velocidad hardware y paralelizar tareas de inyección.

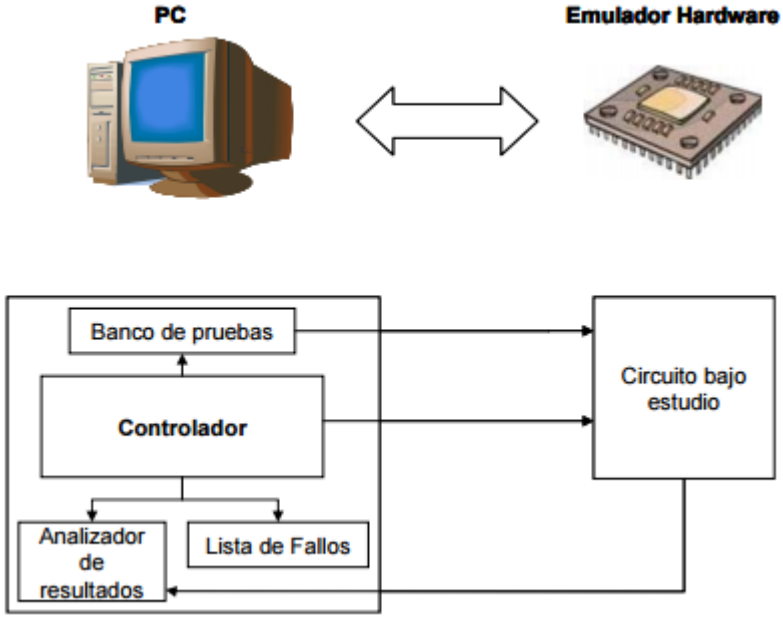


Figura 11. Arquitectura típica de una emulación de inyección de fallos [8].

En función de cómo se realiza la inyección, las técnicas se clasifican en:

- Inyección mediante reconfiguración de la FPGA, modificando el valor del contenido del elemento de memoria a evaluar cada vez que se inyecta un fallo. Requieren la reconfiguración parcial o total de la FPGA para cada fallo.
- Inyección controlada por lógica adicional insertada en el circuito original. Es necesario, por tanto, modificar o instrumentar la descripción original.

Los resultados obtenidos en los distintos trabajos realizados hasta el momento, muestran que las técnicas de inyección basadas en la instrumentación del circuito son

más eficaces en cuestión de velocidad que las técnicas basadas en reconfiguración, aunque requieren la modificación del circuito y la adición de lógica extra, esto es, mayores requisitos en área. Debido al aumento de la densidad de integración en los CIs es posible fabricar FPGAs con millones de puertas, incrementándose la capacidad de procesado de las FPGAs. Esto ha propiciado el avance en el estudio de los métodos basados en emulación, siendo ahora suficiente una FPGA comercial para circuitos relativamente grandes.

En esta tesis se utiliza un entorno de emulación que minimiza la comunicación necesaria entre el PC y la FPGA [8]. La emulación autónoma es una herramienta para la evaluación de la sensibilidad de los circuitos digitales a través de la inyección de fallos transitorios en los elementos de memoria. Ofrece muy buenas tasas de inyección de fallos y permite una completa inyección de fallos (en cada elemento de memoria en cada ciclo de reloj del banco de pruebas), proporcionando no sólo los resultados representativos de sección transversal, sino también la localización de las áreas más sensibles en el circuito para ser endurecidas.

2.3.3. Inyección de fallos con laser

También es posible emular los efectos de la radiación sobre un dispositivo electrónico en un ambiente de radiación espacial mediante la acción de un láser y con sus parámetros de irradiación bien definidos. La inyección de fallos transitorios mediante un haz láser consiste en bombardear el circuito con dicho haz, de forma que la energía de la radiación láser genere pares electrón-hueco, dando lugar a fallos transitorios [39]. En primer lugar, la inyección láser se aplicó para generar fallos permanentes, pero actualmente es una técnica muy utilizada para la inserción de fallos transitorios SEEs.

2.3.4. Inyección de fallos mediante aceleración de partículas

La principal fuente de fallos SEU son las partículas que conforman la radiación ionizante proveniente del espacio. Por lo tanto, la forma más precisa y realista de probar experimentalmente el comportamiento del circuito bajo fallos es someterlo a dicha irradiación. Estos experimentos, denominados test de irradiación, consisten en bombardear el circuito de estudio con un haz de partículas (iones, protones o neutrones). El comportamiento del circuito bajo irradiación se compara con un circuito libre de fallos para detectar los efectos de los fallos. Los fallos transitorios, que se producen como consecuencia de radiar un CI, pueden afectar a cualquier elemento del circuito.

La metodología de inyección de fallos mediante aceleración de partículas consiste en realizar un test estático o dinámico con irradiación de partículas. Los datos obtenidos permiten medir la sección eficaz de un circuito, frente a SEUs, proporcionando información sobre la sensibilidad de dicho dispositivo a este tipo de partículas. Además, conociendo el flujo de partículas del haz, es posible predecir la tasa de fallos SEU que se producirá cuando el circuito esté funcionando normalmente. Este método se utiliza también para comparar la sensibilidad de distintas tecnologías y estudiar los factores que pueden influir en la tasa de fallos, como por ejemplo la frecuencia de funcionamiento. Por otra parte, el test de radiación, al igual que otros métodos de inyección de fallos físicos, puede provocar un fallo que dañe el circuito. Por ejemplo, un problema a considerar cuando se realiza un test de radiación es el efecto de *Latch-Up*, SEL. Este efecto, da lugar a grandes corrientes y a una excesiva disipación de calor que puede dañar el circuito. Puesto que los fallos generados mediante radiación no son controlables, hay que incluir un mecanismo de protección frente a un posible SEL. Este mecanismo consiste en monitorizar la corriente del circuito para detectar cualquier

incremento. En caso de observar un valor de corriente superior a cierto umbral el dispositivo debe apagarse o resetearse.

2.3.5. Otros

En algunos sistemas como las FPGA y los microprocesadores que presentan estructuras y funciones propias que los caracterizan, se han propuesto métodos de inyección específicos que aprovechan dichas estructuras para realizar la inyección.

La inyección de fallos en la memoria de configuración es un problema propio de las FPGAs. La complejidad de los dispositivos reprogramables hace que la inyección de fallos mediante simulación sea un proceso excesivamente lento. A este hecho hay que añadir que normalmente los fabricantes de estos dispositivos no publican los modelos o descripciones del circuito. Por lo tanto, los métodos de evaluación de la tolerancia a fallos de FPGAs comúnmente utilizados están basados en la inserción de fallos en un componente comercial mediante una de las siguientes técnicas:

1. Inserción de fallos físicos aplicando alguna de las soluciones descritas anteriormente.
2. Introduciendo modificaciones, bit-flip, en la memoria de configuración de la FPGA a evaluar. Para modificar un bit de la memoria de configuración es necesario leerla y volverla a escribir con el valor del bit en cuestión invertido mediante reconfiguración en tiempo de ejecución. Los entornos de inyección, desarrollados hasta el momento, dirigidos a las FPGAs se han realizado sobre dispositivos de Xilinx® [46][47] porque este fabricante proporciona la capacidad de leer la memoria de configuración o bitstream en cualquier momento (*readback*), así como el *software* necesario para manipularlo.

2.4. SOLUCIONES PARA RESOLVER EL EFECTO DE LOS FALLOS DE SISTEMAS DIGITALES

Es muy típico en sistemas electrónicos digitales dividir la funcionalidad compleja en tareas simples y construir un sistema electrónico distribuido que contiene varios nodos trabajando de forma concurrente. Este esquema se utiliza en diferentes campos desde hace varios años, donde la adquisición de los datos corresponde a los sensores y las tareas de control corresponden a los actuadores, ejecutándose estos al mismo tiempo, con una comunicación remota (por cable o inalámbrica) que conecta cada nodo con el sistema.

La fiabilidad de los sistemas distribuidos se ha venido considerando desde un punto de vista de la gestión ordenada de diferentes tareas, resolviendo los retardos en transmisión de datos y detectando y corrigiendo los efectos del ruido en los datos transmitidos. Los sistemas electrónicos distribuidos trabajando en entornos críticos se enfrentan además con los fallos que llegan por radiación ionizante, que pueden provocar efectos transitorios en la lógica secuencial SEUs y en la lógica combinacional SETs.

Para asegurar la robustez en estos sistemas, se propone una metodología para el endurecimiento óptimo de los circuitos digitales. En este caso, la robustez de todo el sistema puede ser más alta que la suma de las de cada nodo que compone la red. Por lo tanto, es posible obtener un sistema tolerante a fallos sin endurecer completamente los elementos básicos, pero es crucial localizar correctamente las tareas críticas y los elementos débiles de sistema, para endurecerlos de la forma más eficiente y económica posible.

En los sistemas electrónicos distribuidos compuestos por varios nodos trabajando en concurrencia, obteniendo la información de los sensores y enviando comandos a los

motores y actuadores, el análisis de la sensibilidad de todo el sistema normalmente no se realiza de forma experimental. Esto se debe a que los *tests* de fiabilidad del sistema distribuido completo son difíciles de planificar y de realizar. Por este motivo, se proponen soluciones para ejecutar *tests* fiables a los sistemas distribuidos que ayuden a los diseñadores y a las compañías a localizar los elementos críticos a endurecer, asegurando que el efecto de cada uno de los fallos ha sido analizado, clasificado y gestionado. Actualmente, hay pocos métodos que proporcionen soluciones en este campo [41][42].

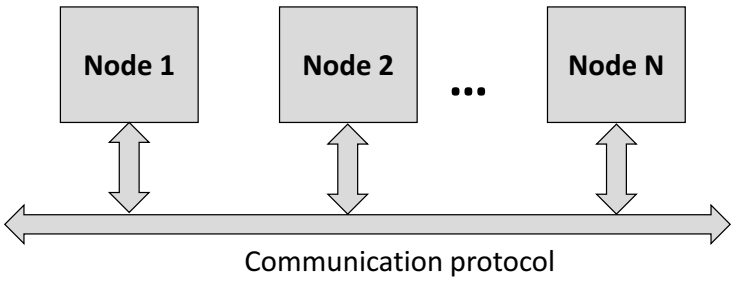


Figura 12. El protocolo de comunicación.

Las técnicas presentadas en los apartados anteriores, están dedicadas a endurecer los componentes electrónicos digitales, particularmente circuitos integrados de aplicación específica (ASIC, *Application Specific Integrated Circuit*) pero también FPGAs (*Field Programmable Gate Arrays*). Los circuitos actuales tienden a la integración de distintos componentes en un mismo chip, SoC, y a la extensión del uso de circuitos lógicos programables como las FPGAs o los SoPCs, proporcionando altas prestaciones con altas densidades de integración. De esta forma, los diseños actuales, cada vez más complejos, presentan nuevos retos en tolerancia a fallos que requieren el desarrollo y propuesta de nuevas técnicas y soluciones para el diseño de circuitos tolerantes a fallos.

2.4.1. Detección de fallos, elección y verificación de los componentes a nivel de sistema.

En un sistema digital distribuido la detección de fallos transitorios, debido a radiación ionizante, así como su mitigación es un problema que puede ser optimizado y organizado de forma más eficiente que los métodos usados en la actualidad.

Tanto el impacto funcional de un SEE en un sistema de estas características, como la probabilidad de su aparición requieren que los diseñadores cumplan varios requisitos del diseño del sistema. Como punto de partida, se especifica la máxima cantidad de SEEs permitida para cada categoría de sistemas y componentes. Las restricciones sobre la cantidad máxima de SEEs admitidos en el nivel de sistema se pueden satisfacer mediante el uso de varias técnicas de mitigación, incluyendo técnicas de redundancia *hardware*, *software*. El enfoque más rentable puede ser una combinación apropiada de los dispositivos *rad-hard* y otras técnicas de mitigación. Sin embargo, la disponibilidad, la potencia, el área, el coste o el rendimiento de los dispositivos *rad-hard* pueden prohibir su uso. La complejidad del sistema puede ser un problema a la hora de utilizar la redundancia *software software* o el *hardware*. Una combinación de los dos métodos puede ser la opción adecuada y efectiva si se utilizan dispositivos no endurecidos.

Los diseñadores de los vehículos espaciales están utilizando cada vez más dispositivos comerciales C*ommercial Off-the-shelf Technologies* (COTS), FPGAs, microprocesadores de alto rendimiento, circuitos integrados de bajo consumo, etc. La razón de esta tendencia está en el interés de combinar el rendimiento, tamaño y coste de

los dispositivos comerciales con las necesidades de procesamiento de las naves espaciales. Las ventajas de este tipo de tecnología son la mayor densidad de área (más puertas para el mismo peso y/o área), aumento de la velocidad y del rendimiento y disminución de los tiempos de entrega en comparación con los dispositivos con la tecnología *rad-hard*. Los fabricantes de circuitos integrados abastecen un mercado comercial donde la comunidad espacial es una parte muy pequeña, por lo que las tecnologías robustas son más caras y más lentas de fabricar. El uso de componentes comerciales en aplicaciones aeroespaciales, hace que estas tecnologías deban ser evaluadas para cumplir con los requisitos de robustez y seguridad de las naves espaciales, especialmente en satélites pequeños de bajo presupuesto donde su uso mucho mayor.

La verificación es el proceso en el que se muestra si el diseño cumple con los requisitos dados. La verificación dinámica se refiere a la comprobación del funcionamiento del circuito cuando está en operación. En la fase inicial del diseño, se realizan una lista de los dispositivos electrónicos necesarios para la construcción del sistema. Esta lista se aprueba por los ingenieros y los expertos en radiación. Los dispositivos que no tienen especificaciones de tolerancia a la radiación deben ser cambiados por otros ya calificados o, si no existe alternativa, se deben realizar pruebas de calificación de estos dispositivos frente a la radiación. Las pruebas de irradiación de los dispositivos electrónicos es la clave en el caso de que no se conozca su tolerancia, reducen el riesgo, el tiempo, el coste de rediseño de las soluciones alternativas.

Los sistemas distribuidos digitales abarcan campos muy diversos como por ejemplo la manipulación de los datos y el control: microprocesadores y microcontroladores (por ejemplo, 8051, LEON, ERC32), los buses de comunicación (AMBA, PCI, 1553, SpaceWire, CAN, LIN), protocolos de comunicación CCSDS / ECSS (telecomando y telemetría), DSPs para procesamiento de imágenes, sistemas de navegación, detectores de radiación, sensores de imagen y muchos más. Todos estos componentes requieren un alto nivel de tolerancia a fallos de todos los componentes por separado y sobre todo del sistema completo. A continuación se enumeran las partes más críticas de un sistema distribuido y los métodos que se utilizan hoy en día para endurecerlos.

2.4.1.1. Enlaces (*links*)

En los sistemas distribuidos se pueden utilizar distintos buses de comunicación, dependiendo de la aplicación y de que tareas tiene que ejecutar el sistema. Uno de los buses de comunicación que se utiliza en la aviónica, el bus de datos MIL-STD-1553B es un ejemplo muy didáctico de un bus que tiene que ser robusto y seguro.

Las técnicas para la detección de los fallos de este protocolo son el bit de paridad, la suma de comprobación y el CRC. La recuperación de la comunicación corresponde a las capas del modelo OSI de niveles superiores (por ejemplo el software en el nivel de aplicación), y se elige la mejor estrategia que se adapte a la aplicación en cuestión, normalmente se realiza la retransmisión del mensaje.

Otros componentes muy importantes en sistemas distribuidos son los sensores. Para un sensor inteligente se utiliza las mismas características que para el bus de aviónica. Para un sensor simple se utiliza la triplicación del sensor (por ejemplo, todos los termistores en el satélite SPOT CNES [12]) y la redundancia temporal.

Para la protección de los actuadores se utiliza el concepto de direccionamiento de las señales enviadas por un camino distinto, uniéndolas mediante una puerta lógica AND. Está técnica normalmente se utiliza en los elementos de pirotecnia.

Otro ejemplo de un enlace serie de alta velocidad es HSSL (*High Speed Serial Link*) se utiliza para los datos de imagen donde los reintentos no están permitidos (demasiados datos para la memoria). Por lo tanto, la estrategia habitual consiste en seleccionar el diseño HSSL que tiene un BER (*Bit Error Rate*) muy bajo, y se realiza su implementación sin protección.

2.4.1.2. Memorias

La confiabilidad de las memorias RAM frente a los errores, es suficientemente alta para aplicaciones domésticas pero para usos más críticos se aplican técnicas de corrección y detección de errores basadas en diferentes estrategias tales como códigos de paridad, EDAC (*Error Detection And Correction*), Hamming descritas en el capítulo 2. Por lo general los sistemas con cualquier tipo de protección contra errores tienen un coste más alto. Para tener un sistema con ECC o paridad, el chipset y las memorias deben tener el soporte para estas tecnologías.

2.4.1.3. Unidades de procesamiento

Las técnicas básicas que utilizan los diseñadores para una arquitectura de microprocesador tolerante a fallos son:

- Replicación de los tiempos a nivel de instrucciones (*Time-TMR SPACE MICRO*). Es una técnica sin necesidad de replicación del hardware y sin coste adicional. El mismo software se procesa N-veces sucesivamente en la misma CPU[110].
- Replicación del tiempo a nivel de tarea.
- Duplicación del sistema. Es una técnica que permite la detección pero no la recuperación del fallo. Las dos condiciones principales para la recuperación son:
    - La memoria se considera libre de los SEEs, gracias al EDAC.
    - El microprocesador estropeado no debe de ser capaz de escribir erróneamente en la zona de memoria donde se almacenan todos los datos. Esto se realiza por medio de la comparación de todos los resultados, y sólo si el 100% de los resultados son correctos.
- TMR-Triple y QMR-Quadruple. Los ejemplos representativos son SHUTTLE, GUARDS, ATV. La detección se realiza por una votación mayoritaria. La recuperación se realiza por el enmascaramiento del fallo, donde los resultados del canal defectuoso se enmascaran de forma continua gracias a los datos emitidos por los canales correctos.
- Triplicación del microprocesador sincronizado. Por ejemplo el SCS750 (SCS750 - *Super Computer for Space*) de MAXWELL Tech. Todos los microprocesadores ejecutan la misma instrucción en el mismo ciclo de reloj. Se requiere que uno de los microprocesadores tenga la capacidad de bloqueo-paso (por ejemplo, la sincronización de los generadores de reloj interno, comparadores de los buses, etc.). La norma de seguridad para automóviles ISO26262, establece las especificaciones para HW y SW en sistemas de control electrónicos administrando los riesgos de eventos peligrosos. Por

ejemplo el micro ARM Cortex-R está orientado a trabajar en tiempo real para sistemas embebidos:

  - Con un enfoque de respuesta rápida y determinista de las interrupciones y de la seguridad / fiabilidad mediante la unidad de protección de memoria, ECC, paridad, bloqueo de paso (*lock-step*).
  - Con un microprocesador de doble núcleo que permite la implementación de una configuración de bloqueo de paso para facilitar el cumplimiento de la norma ISO26262. Esto mismo lo cumplen los microprocesadores de Texas Instruments, LSI, Infineon, Fujitsu, Toshiba.
  - En cambio, el rendimiento de procesamiento de la familia ARM Cortex-R es significativamente más baja que la de otras familias de microprocesadores, por ejemplo la familia PowerPC.

- Arquitecturas tolerantes a fallos *trade-off*. No existe una solución universal, aunque algunos requisitos comunes sean el consumo de energía o el peso del equipo. Por ejemplo, si se está diseñando un equipo COTS para una carga útil de un micro-satélite de 100 kg se podría elegir una arquitectura de replicación temporal. En cambio, si se está diseñando un equipo con componentes COTS para una nave espacial tripulada por humanos, como SHUTTLE, se podría elegir, por ejemplo, una arquitectura 4-MR (Quadruplex) con capacidad de enmascaramiento que se adapta muy bien a la brevedad de las fases críticas. Si estamos diseñando un equipo con componentes COTS para una carga útil de una gran misión científica (1,000 kg) podríamos elegir, por ejemplo, una arquitectura con duplicación con / sin la capacidad de recuperación o una triplicación del microprocesador sincronizado.
- Otros métodos y mecanismos de protección podrían ser:
  - ABFT – Algoritomos basados en tolerancia a fallos.
  - WDP – Procesador WatchDog con análisis de firma.
  - Wrappers.
  - Mecanismos de protección mixtos.

2.4.2. Ejemplos de los sistemas tolerantes a fallos en las aplicaciones reales

Los sistemas electrónicos distribuidos (SEDs) se pueden encontrar en muchos campos, son cada vez más complejos y ejecutan cada vez más tareas. Los SEDs se pueden clasificarse dependiendo de su grado de complejidad, en función de tiempo de funcionamiento, del tamaño, del tipo de comunicación que utilizan, etc.

2.4.2.1. *Sistemas automóviles*

La industria del automóvil es probablemente el área con mayor utilización de los sistemas distribuidos, donde se utiliza la implementación de los sistemas electrónicos de control no sólo para tareas de confort, sino también para tareas que requieren alta seguridad en su operación.

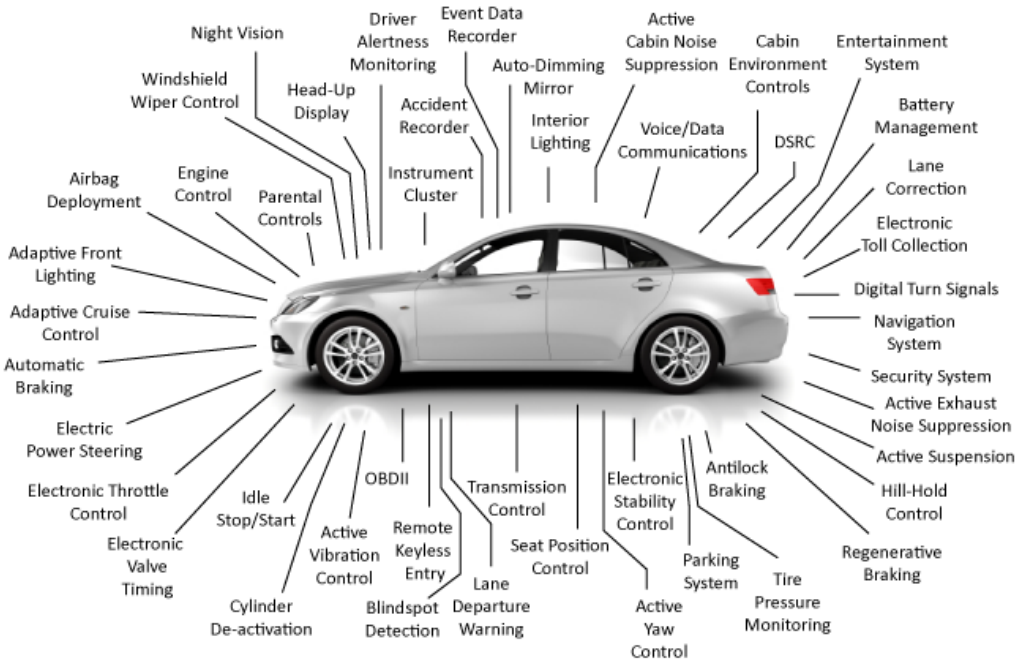


Figura 13. Algunas de las tares típicas de un coche hoy en día [66].

Hoy en día un coche moderno tiene más de 100 unidades ECUs. Por ejemplo un Lexus LS460, cuando todo su equipamiento está instalado, tiene más de 100 ECUs embebidos y cerca de 7.000.000 líneas de código software embebido. Las ECUs de los distintos sistemas tienen diferentes requisitos y requiere distintas tecnologías. La mayoría de las ECUs se usan para los siguientes propósitos:

- Ahorro de energía y reducción de emisiones.
- Seguridad (activa y pasiva).
- Confort, comodidad y entretenimiento.
- Reducción de coste y peso.

A continuación se describe la clasificación de los sistemas de automoción:

Sistema	***Descripción***	***Requisitos de red***	***Protocolo de comunicación***
Control de potencia y chasis	Motor, caja de cambios, frenos, suspensión, direccionamiento	-tiempo de respuesta corto y garantizado -tamaño pequeño de datos -alta fiabilidad	high-speed CAN, FlexRay
Control Electrónico	Panel de instrumentos, llave, puertas, ventanas, iluminación	-gran número de nodos de la red y de los datos -moderada fiabilidad, bajo consumo de energía	low-speed CAN, LIN
Multimedia	Audio, navegación, información de trafico	-gran ancho de banda para los datos multimedia -fiabilidad moderada	MOST, IDB-1394
Los sistemas/servicios integrados	Control electrónico de estabilidad, ABS (*Anti-lock Breaking System*) seguridad de colisión, airbag, aparcamiento asistido, etc.	Alta fiabilidad	

Tabla 4. Clasificación de los sistemas de automoción.

Un ejemplo de un sistema distribuido electrónico podría ser un sistema de control de motor. La función básica de este control es calcular el volumen de la inyección de combustible, el tiempo de encendido y el control de los actuadores en cada ciclo de rotación. Los componentes principales de este sistema son el ordenador de control, numerosos sensores (sensor de posición de cigüeñal, medidor de flujo de aire, sensor de temperatura de admisión, sensor de acelerador) y los actuadores. Para realizar este control, la tarea del tiempo real del sistema es necesaria debido al cálculo del volumen de inyección de combustible que debe estar realizado antes de terminar el tiempo de inyección del combustible. Los requisitos de seguridad también son necesarios. Los fallos en la inyección de combustible no deben ocurrir, debido a que el gas inflamable puede emitirse fuera del motor y provocar un incendio.

2.4.2.2. *ESA Automated Transfer Vehicle* (ATV)

Un vehículo automatizado de transferencia o ATV es una clase de nave espacial robótica que asume las funciones de abastecimiento, la retirada de residuos y la elevación periódica de la Estación Espacial Internacional (ISS). Este vehículo pertenece a la ESA y es fabricado por 38 compañías, siendo EADS el principal contratista. El sistema se ha lanzado a la estación en el año 2008. El vehículo tiene triplicado el principal sistema de monitorización y la computadora de control (FTC: *Fault-Tolerant Computer*). La computadora de chequeo esta duplicada, con esta computadora el vehículo está realizando monitorización de la fase de acoplamiento del vehículo para evitar las colisiones. Está implementada la tolerancia para los errores de software.

2.4.2.3. MYRIADE

Myriade es una plataforma para los satélites en forma de cubo, cuyo lado mide 60 cm, y pesa entre 100 y 150 kg. El sistema de control que tiene implementado es un transputer T805 con 1 Gigabit de memoria y capacidad de cómputo de 5 MIPS. El microprocesador está protegido ante protones con una capa protectora de tungsteno de 2 mm. Está interconectado a través del bus I2C con una FPGA (con registros críticos implementados con una estructura TMR) y tiene un microcontrolador PIC que controla el equipo de a bordo. La plataforma está desarrollada por CNES [12] y en esta se usan mayoritariamente los circuitos COTS.

Los circuitos integrados sensibles se desconectan automáticamente en caso de no estar utilizados. Es una técnica que mejora el TID de los componentes en las condiciones extremas. Para los fallos SEL están implementadas unas resistencias en serie con las pistas de alimentación o un limitador de corriente. Para los fallos SETs, está implementado el filtrado de adquisiciones analógicas y la redundancia temporal. Para los SEUs la protección de los buses de datos es realizada por medio de checksums y CRC y protocolos de recuperación. Por otro lado, las memorias Flash y FRAM están protegidas con redundancias en los datos, checksums o CRC. Estas memorias se apagan después del arranque del software de vuelo. Se aplica el TMR para los datos críticos almacenados en la memoria del transputer. También se realiza la monitorización de algunos registros internos críticos de microprocesador (temporizadores, etc). El watchdog (WD) está implementado en varios niveles (PIC, en entradas/salidas, Transputer, etc.).

Finalmente cabe de destacar que Myriade es un ejemplo típico de un ordenador desarrollado con componentes comerciales y protegido por una mezcla de mecanismos para una misión concreta.

2.4.2.4. *Solar Orbiter*

Habitualmente, en las aplicaciones reales, debido a su complejidad, es necesario utilizar varias técnicas de tolerancia a fallos conjuntamente, alcanzando un compromiso entre el coste en área, prestaciones y el nivel de confiabilidad. Por ejemplo, la misión de ESA *Solar Orbiter* cuyo lanzamiento está previsto para el julio de 2017 tiene un sistema de 21 sensores para proteger el sistema contra el entorno radioactivo del Sol. La nave espacial tiene múltiples sistemas a manejar, y uno de los subsistemas soporta simultáneamente tareas de telemetría, telecomando y navegación.

Es otro ejemplo ilustrativo, que presenta las técnicas de mitigación de SEUs que se han aplicado a las diferentes generaciones de microprocesadores para aplicaciones espaciales desarrolladas por la agencia espacial europea (ESA, *European Space Agency*). En concreto, el microprocesador LEON2-FT incluye TMR en los biestables, códigos de detección y corrección de errores en el fichero de registros, código de paridad en las memorias caché y la triplicación del árbol de reloj con cierta desviación (*skew*) para evitar SETs [49][50].

2.4.2.5. *OPTOS*

En general, los satélites están equipados con numerosos sensores para cubrir las diferentes funciones como pueden ser la actividad científica de la misión, las tareas de monitorización y mantenimiento, la determinación y control de la posición. Actualmente, se utiliza la comunicación inalámbrica dentro del satélite, para minimizar peso y consumo, como por ejemplo en el caso del Instituto Nacional de Técnica Aeroespacial (INTA) que ha lanzado un *cubesat*, el satélite OPTOS con comunicación interna óptica. El concepto del computador de OPTOS es el de un diseño que responde a una arquitectura modular y distribuida. Esto implica que el OBDH (*On Board Data Handing*) está formado por diversas unidades con distinto grado de inteligencia y repartidas por el todo el satélite. De este modo, por un lado se da respuesta a las necesidades de cada subsistema (comunicaciones, ADCS, potencia, carga útil, etc.) y por otro mantiene la funcionalidad propia del computador: tiempo del sistema, *housekeeping*, supervisión, gestión de interrupciones, etc. Además de los servicios citados, el computador pretende ser una solución de bajo gasto y consumo manteniendo por un lado las altas prestaciones, materializadas en la velocidad de cómputo (arquitectura RISC de 32 bits y 48MHz de reloj), el manejo de memoria externa e interna (4 Gbits de memoria FLASH y 8 Mbits de RAM), etc., y por otro lado aportando máximos niveles de fiabilidad y calidad que se exigen en los sistemas espaciales [65].

El OBDH de OPTOS está formado por dos tipos de unidades:

- El EPH (*Enhanced Processing Hardware*). Es el núcleo del sistema y está basado en un procesador software (SoPC) implementado en una FPGA y está asociado principalmente a la gestión del subsistema de comunicaciones con tierra.
- Unidades DOT (*Distributed On board computer Terminal*). Son pequeños dispositivos programables (CPLD Coolrunner-II) capaces de implementar funciones simples orientadas a comandar y gestionar de forma autónoma los distintos subsistemas (experimentos científicos) y unidades del satélite.

El nivel de calidad aplicable a las CPLD de las DOT y a la FPGA del EPH es MIL-883 con el fin de asegurar que van a soportar las condiciones de temperatura y vacío impuestas por la misión. No obstante, desde el punto de vista del ambiente radiactivo (dosis y partículas ionizantes) los componentes deben ser también aptos para

la misión, por lo que se ha desarrollado un programa de caracterización del comportamiento en condiciones de espacio para aquellos circuitos sin datos específicos por parte del fabricante.

En el caso de la FPGA, ha sido elegida una VIRTEX II Q-pro de Xilinx que soporta hasta 50 krads y está libre de *LatchUps*. Sin embargo, es bastante sensible al impacto de partículas (protones e iones pesados) de energías por encima de 1 MeV. Las CPLD que se utilizan como núcleo de las DOT también presentan una alta sensibilidad frente a partículas ionizantes, pero un buen comportamiento frente a dosis acumulada (>30 krads).

El resultado del *test* de irradiación, nos ayudó a predecir errores funcionales en la memoria de configuración de las CPLD del orden de uno cada 40 días, es decir, resulta necesario reconfigurar el sistema periódicamente. Esto se consigue reiniciando las unidades cada cierto tiempo aprovechando la redundancia implícita de la configuración y manteniendo el requisito de que al menos tres nodos (DOTs o EPH) estén siempre encendidos. Esto permite conseguir un 100% de fiabilidad y además se reduce el consumo medio al 10% respecto al de un sistema siempre encendido.

Capítulo 3 MEDIDA DE LA SENSIBILIDAD A LA RADIACIÓN IONIZANTE EN SISTEMAS DISTRIBUIDOS DIGITALES Y SUS ENLACES DE COMUNICACIÓN

Los sistemas distribuidos digitales se usan extensamente en aplicaciones que ejecutan un gran número de tareas diferentes y que constan muchos elementos de control, incluso en modo remoto (red distribuida). Las industrias aeroespaciales y del automóvil destacan en el uso de sistemas distribuidos, debido a que estos sistemas son muy eficientes para ejecutar varias tareas a la vez con bajo coste, bajo consumo y una alta rentabilidad. Existen varios estándares de comunicaciones para sistemas distribuidos, que se diferencian fundamentalmente por el sector donde se va a usar el sistema. Estos estándares definen principalmente las tareas de distribución y de comunicación de la red.

Cuando un sistema distribuido trabaja en condiciones muy extremas, como por ejemplo un entorno de radiación ionizante, son necesarias técnicas de mitigación de errores, que aseguren el funcionamiento a lo largo de un tiempo mínimo. Normalmente estas técnicas están basadas en el concepto de redundancia, explicado anteriormente. Pero cuando el elemento que tiene que ser endurecido no es una simple tarea en un único componente sino una tarea distribuida entre varios nodos, las técnicas tradicionales no aportan soluciones eficientes.

3.1. ROBUSTEZ DE LOS SISTEMAS DISTRIBUIDOS

Como ya se ha mencionado, en los últimos años, los sistemas distribuidos para aplicaciones embarcadas (funcionamiento autónomo) han extendido su uso a muchos campos, mayoritariamente la automoción, la aviónica y la industria aeroespacial. El crecimiento de la complejidad de estos sistemas y especialmente su robustez imponen a los ingenieros la búsqueda de nuevas soluciones y herramientas para el proceso del diseño y desarrollo.

Aunque la funcionalidad distribuida entre los nodos individuales ha sido muy utilizada en el pasado, la novedad actual es el tipo de nodos y las técnicas de endurecimiento del sistema, utilizados para construir los sistemas distribuidos. Por ejemplo, en caso de un error crítico los satélites tienen alarmas que activan el sistema de recuperación.

Actualmente están disponibles, para los sistemas distribuidos, nodos muy potentes computacionalmente, con un consumo de energía y un coste muy bajos, gracias a la evolución tecnológica. Estos nodos son capaces de ejecutar, al mismo tiempo, tareas locales muy complejas y algunas tareas globales críticas, con valores de rendimiento muy interesantes en términos de velocidad, el área y el coste.

Como se ha mencionado anteriormente en el caso más general, el proceso de endurecimiento de este tipo de sistemas implica utilizar las tareas con redundancia ya sea en hardware, en software o en bloques de vigilancia. En los sistemas distribuidos electrónicos, utilizados en los coches, aviones o naves espaciales, nos enfrentamos con tres problemas principales. En primer lugar, la distribución de las tareas entre los

diferentes nodos de la red debería implicar un correcto grado de modularidad, para minimizar las comunicaciones que resultan ser un factor de posibles retardos. En segundo lugar, la ejecución distribuida de tareas críticas debería implicar un alto grado de robustez a nivel de sistema. Por último, el enlace de comunicación debe asegurar un bajo nivel de propagación de errores.

Asumiendo una comunicación robusta, las tareas distribuidas y las técnicas de mitigación de errores deben considerarse cuidadosamente. En la bibliografía no se ha encontrado un método propuesto para ayudar en el proceso de diseño y desarrollo de sistemas distribuidos robustos, que permita comprobar, a nivel de sistema, la eficacia de las técnicas de reducción de fallos y averiguar las áreas o tareas más críticas al efecto de los fallos.

3.2. METODO DE EVALUACIÓN DE SENSIBILIDAD DE LOS SISTEMAS DISTRIBUIDOS FRENTE A SINGLE BIT UPSET

Actualmente, la evaluación de la sensibilidad SBU se puede realizar antes de la fabricación de los circuitos por medio de la simulación o de la emulación (prototipando el hardware del diseño del circuito con la capacidad de auto-inyección de fallos de acuerdo con un modelo de fallo, por lo general un bit-flip)[8]. Las técnicas de emulación son aceptadas, como la mejor solución en la fase de diseño, dadas sus grandes tasas de inyección de fallos y su gran capacidad de procesamiento.

Una vez fabricado el circuito, la sensibilidad SBU / SEU se testea por medio de inyección de partículas o experimentos de inyección con el láser. Estos tests son imprescindibles para calificar los circuitos y sistemas digitales.

La evaluación de la sensibilidad de SBU / SEU de un circuito endurecido con técnicas de mitigación de fallos implica un análisis de detección de posibles fallos y capacidades de recuperación del funcionamiento del sistema después de la inyección. Esta evaluación debería hacerse antes de la fabricación del circuito, con el fin de seleccionar la técnica de mitigación más adecuada [68]. La forma más sencilla de medir la sensibilidad de los circuitos es clasificando los fallos según su efecto. La clasificación de los efectos de los fallos más general es la siguiente [8]:

- Avería (*Failure*) cuando la inyección de un fallo provoca un funcionamiento erróneo del sistema.
- Silencioso (*Silent*) cuando el efecto desaparece por completo sin producir ningun efecto, porque un mecanismo de tolerancia a fallos lo ha enmascarado (por ejemplo una estructura TMR cuando el fallo se produce en uno de los biestables que lo forma) o bien por la propia actividad del sistema (como por ejemplo ocurre cuando un fallo se inyecta en un biestable justo en el instante anterior a que se escriba un nuevo valor).
- Latente (*Latent*) cuando el efecto permanece almacenado en el circuito pero no ha provocado ninguna avería durante el tiempo que dura el experimento de inyección. Este tipo de efectos deben ser estudiados con atención porque pueden dar lugar a un mal funcionamiento si la ejecución continúa.
- El fallo se clasifica como detectado (*Detected*) cuando existe un mecanismo de tolerancia a fallos que detecta el error, generalmente, con el objetivo de activar la recuperación del circuito de dicho de fallo.

Esta clasificación puede ser modificada con el objetivo de obtener más información o por el contrario resumirla. Por ejemplo, en microprocesadores es

interesante distinguir entre distintos tipos de averías, diferenciándose entre errores en las salidas o pérdidas de la secuencia de ejecución [8]. Por otro lado, en algunos circuitos con escasa observabilidad no es viable comprobar si el fallo permanece almacenado o no en el circuito para distinguir entre fallos latentes y silenciosos, por lo que la clasificación se reduce a diferenciar entre aquellos fallos que causan averías y los que no. Pero, cuando el sistema bajo análisis presenta una distribución de algunas tareas entre los diferentes nodos, el análisis de fallos pasa a ser mucho más complicado. Las tareas se ejecutan de forma redundante en cada nodo, y los resultados de las tareas locales solo se pueden evaluar localmente y corregir en el proceso de funcionamiento. Además, cuando las técnicas de mitigación incluyen la distribución de las tareas críticas, las acciones de recuperación de errores son necesarias. Las clasificaciones primeras y siguientes de un fallo, así como de detección y corrección de latencias, son datos importantes para la elección de la mejor técnica de mitigación para un sistema distribuido, donde otros nodos están ejecutando las mismas tareas críticas al mismo tiempo [52].

El método propuesto para la evaluación de la sensibilidad de los sistemas distribuidos consiste en una aplicación software (para una configuración de la campaña de inyección de fallos) y una plataforma hardware (para la realización de la campaña de inyección de fallos, la clasificación de los fallos y el análisis del efecto de los fallos).

Para realizar el análisis de los SBUs se ha utilizado un entorno de inyección de fallos *bit-flip* basado en emulación, que no requiere la interacción con el PC durante el proceso de evaluación. Este entorno se denomina Sistema de Emulación Autónoma, ha sido desarrollado por Marta Portela García para su tesis doctoral, realizada en el grupo de investigación DMA de la Universidad Carlos III de Madrid, y consiste en implementar el sistema completo de inyección en la FPGA junto con el circuito a comprobar [8].

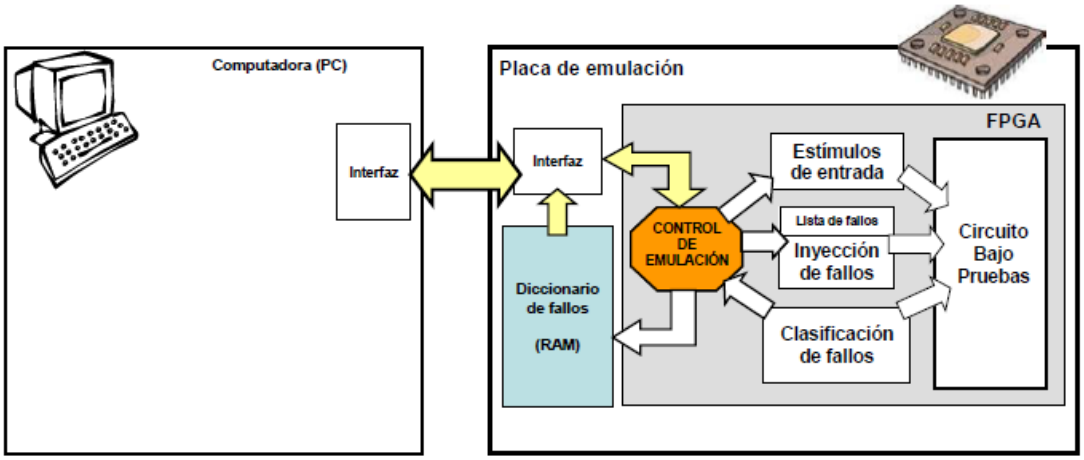


Figura 14. Arquitectura de la Emulación Autónoma [8].

A continuación se describen brevemente todos los módulos implementados en *hardware*:

- **Estímulos de entrada**. Son los vectores de entrada al circuito que se almacenan en una memoria interna o se generan concurrentemente, durante el funcionamiento del circuito. El almacenamiento del banco de pruebas en memoria consume gran cantidad de recursos internos y puede suponer una limitación cuando se trata de grandes cantidades de datos. En este caso, se pueden aplicar técnicas de compresión para reducir la información a almacenar.

- **Clasificación de fallos.** Este bloque representa la función de observación y clasificación de los fallos. El *hardware* necesario para realizar esta tarea depende de la clasificación que se quiera realizar. Para la detección de averías es necesario observar las salidas del circuito y compararlas con las esperadas cuando el circuito se ejercita libre de fallos. En [8] se proponen distintas implementaciones del sistema y por lo tanto distintas soluciones para realizar esta clasificación de fallos.
- **Inyección de fallos.** Este módulo realiza la inyección de fallos. En la Emulación Autónoma se utiliza la instrumentación del circuito como mecanismo de inyección. La implementación *hardware* del control de la inyección mediante la instrumentación del circuito es más sencilla que en el caso de la reconfiguración.
- **Control de emulación.** Este bloque consiste en una máquina de estados que se encarga de controlar el proceso completo de inyección, insertando cada fallo, decidiendo cuando la emulación de un fallo finaliza y es necesario continuar insertando más fallos o por el contrario que la campaña ha terminado, etc.
- **Circuito bajo estudio.** Consiste en el circuito a evaluar modificado con la lógica necesaria para inyectar fallos y observar su comportamiento. A la largo de esta tesis doctoral se han realizado varias emulaciones con distintos sistemas distribuidos a probar.

En la Figura 14 se muestran los bloques necesarios para la realización de estas campañas de inyección de fallos. La adaptación de estos bloques para la inyección de fallos transitorios por emulación en sistemas distribuidos está detallada en la Figura 15.

Aplicar las técnicas de tolerancia a fallos para reducir la sensibilidad de los sistemas digitales es una solución que requiere un compromiso entre los costes del sistema, los recursos y la robustez. Los diseñadores deben decidir qué elementos deben ser protegidos y que nivel de robustez es suficiente para el sistema a desarrollar. Como consecuencia, el compromiso buscado entre coste, robustez y área hace que no todos los elementos están protegidos. Activación de las técnicas de mitigación se corrige o informa sobre los errores más críticos, pero otros permanecen de forma latente en el circuito.

Tradicionalmente múltiples errores se consideran como menos probables y, por lo tanto, las técnicas de mitigación se centran principalmente en los errores simples. A veces consideran generación de varios errores en la misma palabra (una ráfaga de errores) pero siempre causado por un mismo evento.

Dependiendo de la aplicación considerada, el número de partículas que afectan al mismo dispositivo por unidad de tiempo puede ser muy alto. En algunos casos, el dispositivo debe recuperarse de los efectos de una partícula antes del impacto de siguiente partícula. Por ejemplo, para los satélites en órbita LEO (850 km, 90º) se prevé que el número de partículas de tipo protones por centímetro cuadrado, por día es alrededor de 1.58E09 [69]. Mientras que a nivel del mar, el número de partículas de tipo neutrones por centímetro cuadrado y por día es alrededor de 13. Eso significa que casi 2 partículas llegarán al sistema de un centímetro cuadrado, trabajando a 1 MHz, cada 100 ciclos de reloj en una órbita LEO y cada 10^{10} ciclos de reloj a nivel del mar.

El problema es más serio cuando se aplica al esquema de protección de un sistema distribuido con varios nodos (con diferentes tareas específicas) que sejecutan simultáneamente las tareas globales críticos. Cuando los SEUs / SBUs afectan a los

elementos de la memoria en un sistema distribuido, el esquema TMR corrige la salida errónea y el elemento de memoria afectado. Pero, si el fallo afecta a un elemento de memoria sin protección, el error permanecerá latente aunque la salida errónea se corrige. Este error puede causar nuevos errores (corregidos por TMR también). El peor caso es cuando un nuevo fallo afecta a otro nodo, se generan nuevos errores latentes y el sistema se corrige erróneamente.

La presencia de los errores latentes en los sistemas digitales no es inusual y los diseñadores deben tener en cuenta los múltiples errores como la consecuencia de los defectos latentes causados por diferentes SBUs. Por lo tanto para realizar un análisis del sistema distribuido se ha modificado la herramienta de Emulación Autónoma incluyendo la emulación de los múltiples errores y la observación de errores latentes [51], [52].

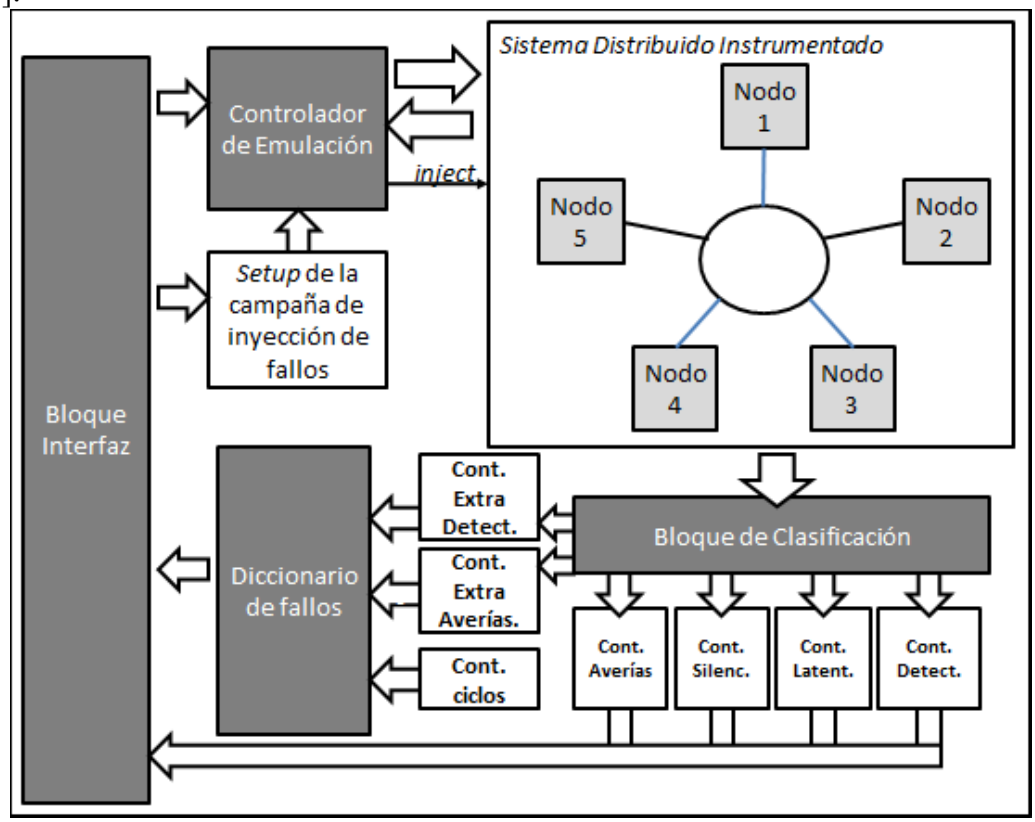


Figura 15. Plataforma hardware para Evaluación de la sensibilidad de los sistemas distribuidos robustos [51].

Para este caso se emulan dos réplicas del sistema: una con fallos insertados (*Faulty*) y una libre de fallos (*Golden*). Esto es igual que para circuitos simples. Lo que cambia es la observación de los efectos de los fallos. Ahora se mira de forma global el efecto perjudicial de los fallos (Averías). Por dos razones, primero porque aunque un nodo esté fallando, si ese fallo corresponde a una tarea global distribuida su efecto no tiene por qué ser avería; segundo, porque si el fallo se corresponde a una tarea local, puede que no sea avería del sistema (lectura de sensor, comando no crítico en actuador, tarea de comunicación endurecida en capas más altas del modelo OSI, etc. En este nuevo sistema de evaluación no se detiene la emulación de un circuito con fallo cuando un fallo se clasifica. Se sigue emulando hasta el final de la carga de trabajo aplicada al sistema, porque la clasificación evoluciona a lo largo de la misma y sí es importante analizar el efecto acumulado de los fallos latentes, especialmente para las tareas globales. Cada vez que se inyecta un fallo se computa cuántas veces se clasifica de cada manera y cuantos cambios de clasificación puede llegar a sufrir durante un ciclo de

trabajo típico. Se permite la inyección de fallos dobles en bloques de memoria cercanos, emulando el efecto de los MBU en tecnologías nanométricas.

Es necesario que durante la campaña de inyección de fallos por emulación en un sistema distribuido, se analice de forma detallada el efecto de los fallos latentes en las tareas globalmente mantenidas por la red distribuida, así como que se detalle de forma más precisa qué se considera avería, teniendo en cuenta que las técnicas de mitigación de fallos globales y locales.

3.3. MÉTODO DE EVALUACIÓN DE LA SENSIBILIDAD A LA RADIACIÓN IONIZANTE DE SISTEMAS DISTRIBUIDOS

El método propuesto en el apartado anterior se ha comprobado mediante su aplicación en un sistema distribuido genérico. Este sistema distribuido está compuesto por un conjunto de nodos básicos, cuya arquitectura ha sido cuidadosamente analizada mediante un estudio de los sistemas propuestos en la literatura para los sectores de automoción y aeroespacial. El nodo básico elegido será el punto de partida para proponer un sistema distribuido, sobre el que aplicar un sistema de evaluación de la sensibilidad mediante emulación *hardware*, sobre el que incluir técnicas de mitigación de fallos y sobre el que incluir estructuras que ayuden a realizar los *tests* de calificación y los *test* en funcionamiento de los sistemas finales.

3.3.1 Nodo Básico y Sistema Distribuido

Para la propuesta de un método de evaluación de la robustez de sistemas distribuidos, se plantea como punto de partida la definición del nodo básico genérico para este tipo de sistemas. En primer lugar, se ha realizado un estudio de la literatura científica y técnica donde se describen este tipo de sistemas.

El artículo [71] propone la utilización de un nodo para un sistema distribuido de micro-aviónica. Estos autores, de la Universidad Politécnica de Cataluña, consideran que el tipo de nodos que están utilizando debe contener "*toda la electrónica necesaria para apoyar las necesidades de interconexión de sensores locales, instrumentos y otros, así como proporcionar el procesamiento local de datos y de almacenamiento, y la comunicación entre nodos*". El nodo propuesto en este artículo es un SoPC que incluye un microcontrolador, bloques de memoria (RAM, NVM), reloj, temporizadores, *whatchdog*, entradas y salidas analógicas y digitales. Este nodo básico es muy similar a otro propuesto y presentado para el sector espacial en el artículo [73].

Los nodos básicos sencillos también se utilizan en los nano-satélites por ejemplo el nodo básico que ha sido desarrollado por la ESA [72]. En el satélite OPTOS [74], desarrollado por INTA (Instituto Nacional de Técnicas Aeroespaciales), y ya descrito en el capítulo anterior, estos nodos se alimentan de forma independiente mediante baterías, la comunicación de los datos es inalámbrica y están construidos con dispositivos reprogramables CPLDs, que presentan un consumo de energía muy bajo (CoolRunner™II de Xilinx®). En este nodo básico se incluyen bloques lógicos programables y bloques de memoria, no realizan tareas muy complejos, por lo que sólo es necesario un microprocesador externo para la distribución y gestión de las tareas locales y globales.

Con respecto a la industria del automóvil, el nodo básico para los sistemas distribuidos de automoción, es muy parecido al nodo básico de micro-aviónica. Muchos autores utilizan el protocolo FlexRay, mientras que otros mantienen protocolo de bus CAN, pero todos ellos se aplican a un nodo básico compuesto por un microprocesador, bloques de memoria y controladores de comunicación.

Este nodo básico incluye un microprocesador, bloques de memoria volátil y no volátil, una lógica programable y un bloque de interfaz para implementar el protocolo de comunicación. La distribución de las tareas entre los recursos lógicos y el microprocesador se decide en función de la aplicación distribuida final. Este nodo está descrito en los artículos [51] [52]. En la Figura 16 se muestra el esquema de este nodo básico propuesto. El microcontrolador puede ser cualquiera, pero para las primeras pruebas se implementó un microcontrolador PIC de 16 bits.

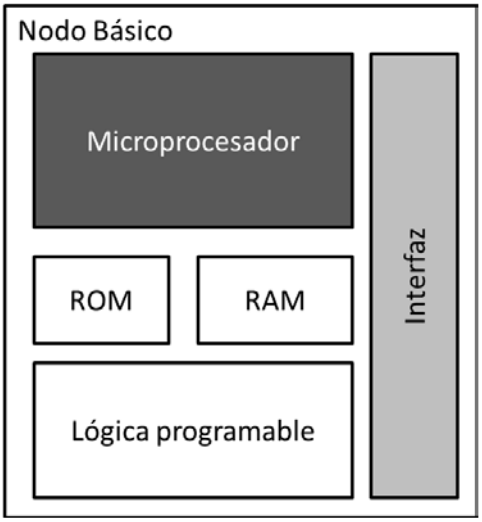


Figura 16. Nodo básico para un Sistema distribuido genérico [51].

Una vez definido el nodo básico, se plantea un sistema distribuido típico, lo suficientemente complejo para ser representativo, pero lo suficientemente sencillo para permitir los distintos análisis de forma rápida y precisa. El sistema distribuido diseñado está compuesto por un número impar de nodos básicos, que pueden estar incluidos en cualquier sistema para vehículos de automoción y aeroespacial. Los nodos ejecutan las tareas locales de acuerdo a la carga que está conectada a ellos, y también gestionan las tareas críticas globales, como por ejemplo el mantenimiento del tiempo real. Por motivos de comparación, en la primera propuesta de sistema distribuido, esta tarea global no sólo está en el *hardware* de la lógica programable, sino también en el *software* del microcontrolador PIC implementada con un temporizador. Para este fin, dos puertos de entrada/salida se utilizan para la cuenta de segundos y de milisegundos. En la figura 17 se muestra el sistema distribuido genérico propuesto en un principio para evaluar su robustez mediante técnicas de inyección de fallos mediante emulación.

En este sistema distribuido propuesto, el tiempo real se mantiene en todos y cada uno de los nodos del sistema y, periódicamente, se comprueba realizando una votación por mayoría, de forma distribuida en el caso de *hardware* y de forma local en el caso del *software*. Para corregir los efectos de los fallos de cada bloque sobre la tarea crítica del tiempo real, se aplica la técnica de redundancia modular triple (TMR). La mitigación del error es inmediata en el caso de la implementación hardware. En el caso del microcontrolador PIC, la interrupción que corrige el contador de tiempo real se genera cuando se activa la acción de corrección y el tiempo correcto se envía por el puerto de salida.

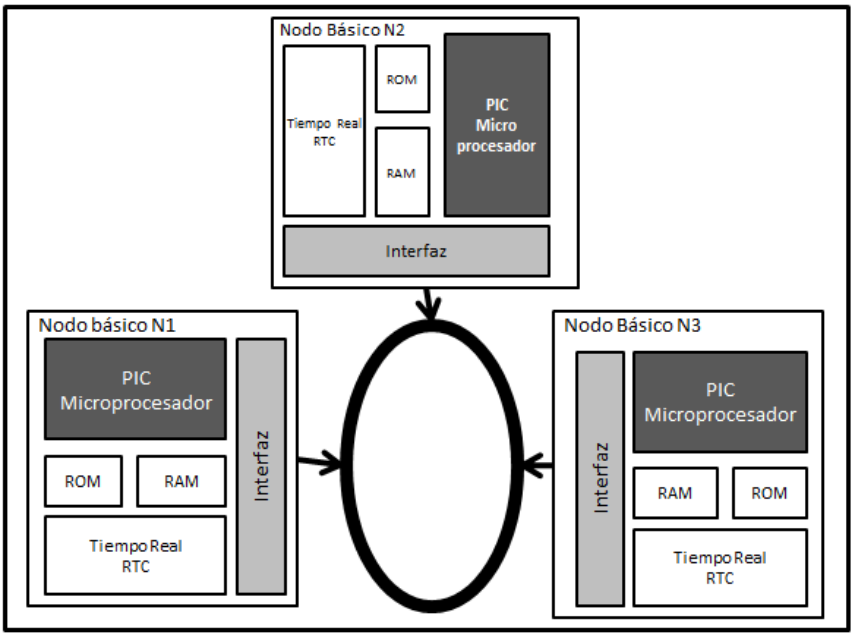


Figura 17. El sistema distribuido genérico [51].

3.3.2. Resultados de la clasificación de los fallos.

El sistema distribuido propuesto e implementado ha sido prototipado en una placa de desarrollo de Digilent®, que incluye una FPGA de Xilinx, Virtex5 (Xc5VLX110T). Aunque la aplicación final tendrá un dispositivo electrónico por nodo del sistema, en la validación del sistema de evaluación de la sensibilidad que se propone en esta tesis doctoral, se prototipa todo el sistema en una única FPGA. La Tabla 5 muestra los datos de ocupación del sistema distribuido genérico. Los datos indicados para los bloques *Contador_RTC* y *PIC* se corresponden con bloques del nodo básico, mientras que la fila titulada *Sistema_Distribuido* corresponde al sistema completo con todos los nodos instrumentados para poder realizar la inyección de fallos. El resto de filas se corresponden con los bloques del sistema de emulación de la inyección y clasificación de los fallos transitorios.

Bloque	**FFs**	**Bloques RAMs** (18K)	**LUTs**
FPGA (recursos disponibles	**69.120**	**296**	**69.120**
Contador RTC	35	0	54
PIC	669	17	2.190
Sistema Distribuido (instrumentado)	10.977	99	20.440
Control de emulación	77	0	213
Bloque interfaz	545	0	835
Bloque Clasificación de fallos	40	0	10
Contadores de clasificación	173	0	160
Diccionario de fallos	0	4	0
TOTAL (recursos utilizados)	**12,017**	**103**	**21,652**

Tabla 5. Resultados de la implementación del sistema distribuido propuesto.

Los fallos inyectados en el bloque RTC afectan a todos los flip-flops involucrados en la tarea de contar el tiempo real. Cada fallo inyectado provoca un valor diferente del tiempo real en el nodo afectado, que activa la protección TMR. Los fallos inyectados en el microcontrolador PIC afectan a los biestables del temporizador, control de la interrupción, los puertos de entrada y salida, y registros de configuración. Cuando un fallo provoca un tiempo real diferente, la protección TMR se activa y se genera una interrupción en el microcontrolador del nodo. Algunos ciclos más tarde, el valor del tiempo real se corrige.

En el método propuesto de evaluación de la sensibilidad en sistemas distribuidos, se realiza un análisis más detallado durante la emulación con el fin de determinar el comportamiento del sistema distribuido después de la inyección de fallos y la mitigación de error. Por lo tanto, la emulación del sistema continúa una vez que el fallo se clasifica por primera vez, se recogen otras clasificaciones y se almacenan.

Bloque	#Fallos	% Silencioso	% Detectado	% Latente	% Avería
RTC	0.297M	0	100	0	0
PIC	2.28M	1	78	21	0

Tabla 6. Primera clasificación de los fallos en el sistema distribuido digital básico.

Las primeras clasificaciones se reportan en la Tabla 6. Esta clasificación se obtiene desde el punto de vista del sistema. No hay averías, sólo fallos detectados, debido a que se han aplicado dos técnicas de protección TMR (*Hardware* y *Software*) se incluyen en el sistema para la tarea del tiempo real del microcontrolador y para el bloque RTC. En la Tabla 6, se muestra la clasificación típica (Avería, Silencioso, Detectado y Latente). Para verificar los sistemas distribuidos, se añade nuevas categorías sobre el efecto de los fallos por emulación, Tabla 7.

- DS significa que el efecto producido había activado una técnica de mitigación, y una vez que la técnica de corrección ha sido aplicada, el fallo desaparece.
- DDL significa que el efecto había activado alguna técnica de mitigación, pero a pesar de que la corrección se aplica, de nuevo se activa la detección de los fallos a lo largo de la emulación hasta el final de la carga de trabajo.
- DL significa que el efecto se corrige por alguna técnica de mitigación, pero hay algunos elementos de la memoria que mantienen el efecto del fallo hasta que termine la carga de trabajo a pesar de que no se activan nuevas detecciones de los fallos.

Las clasificaciones DDL y DL deben ser tenidas en cuenta para garantizar la robustez del sistema. Los fallos que causan estas clasificaciones adicionales podrían provocar un error múltiple simultáneo en los diferentes nodos del sistema. Como era de esperar, los fallos inyectados en el bloque RTC siempre se detectan, mientras que en el microprocesador PIC podrían afectar a los registros de configuración y permanecer latentes durante un tiempo muy largo sin ser propagados a las variables del tiempo real.

Block	%DS	%DDL	%DL	Otros
RTC Contador	66	33	0	0
PIC (Temporizador 0)	0	90	0	1
PIC (Puertos)	50	50	0	0
PIC (Interrupciones)	0	0	7,6	92,3

Tabla 7. Extra Clasificación

3.4. ROBUSTEZ EN LOS ENLACES DE COMUNICACIÓN

Los protocolos de comunicación robustos se utilizan en muchas aplicaciones donde el entorno de trabajo está sufriendo interferencias externas, como la radiación ionizante, interferencias electromagnéticas, interferencias por las fuentes de ruido, etc. Algunos de estos protocolos son propiedad intelectual de la empresa que los desarrolló, mientras que otros están estandarizados por la normativa de ISO o IEEE. Ninguno de los protocolos analizados puede considerarse mejor que otro, tienen características diferentes porque es diferente su campo de aplicación. En estos campos de aplicación deben cumplir con los estrictos requisitos de las normas y reglamentos: en la automoción, la reciente norma es ISO 26262 descrita en el Capítulo 2.

En la Tabla 8 se muestran los siete protocolos utilizados en la industria, incluyendo la empresa o la institución que los desarrolló. En la Tabla 9 se enumeran diferentes características de estos protocolos. Finalmente, en la Tabla 10 se muestran las aplicaciones más típicas para estos protocolos, así como sus características de tolerancia a fallos [69].

Protocolo	***Fabricante y Descripción***
MIL-STD 1553	*El Departamento de Defensa (USA):* Protocolo para bus de comunicaciones serie, utilizado en subsistemas embarcados para el manejo de datos en vehículos espaciales, militares y civiles.
ARINC 429	*Boeing & Aeronautical Radio Inc.:* Protocolo para intercambio de datos dentro de numerosos sistemas de aviónica, de aeronaves comerciales y de transporte.
SpaceWire	*European Space Agency & EECS:* Protocolo para bus de comunicación serie utilizado en vehículos espaciales, basado en el estándar IEEE 1335.
FlexRay	*FlexRay Consortium:* Protocolo de comunicación utilizado en los automóviles.
CAN	*Robert Bosch: Controller Area Network:* Protocolo de comunicación utilizado en los automóviles para aplicaciones críticas.
Ethernet	*Xerox:* Protocolo de comunicación utilizado entre los ordenadores.
LIN	*Local Interconnect Network. Desarrollado por un consorcio de compañías europeas*. Un protocolo barato de transmisión de datos de automoción a distancias cortas.

Tabla 8. Diferentes protocolos de comunicación.

Protocolo (STD)	***Tipo de comunicación***	***Velocidad***	***Medio físico[1]***	***#elementos***
MIL-STD 1553	*Bus Serie Bidireccional*	*1Mb/s 20Mb/s (1773)*	*TSP, OF*	*1 Maestro 31 Esclavos (máx.)*
ARINC 429	*P2P, Serie,Unidireccional*	*12.5kb/s -100kb/s*	*TP*	*1 emite, 20 reciben*
SpaceWire	*P2P, Serie, Bidireccional*	*2Mb/s - 400 Mb/s*	*TP, OF*	*1 Maestro, 1 Esclavo*
FlexRay	*P2P & Bus Serie Bidireccional*	*10Mb/s*	*DP*	*De 22 a 64 nodos*
CAN	*Bus Serie, Bidireccional*	*10kb/s - 1Mb/s*	*TP,OF,OW*	*Multi-maestro(2048)*
Ethernet	*Bus & P2P Serie Bidireccional*	*10Mb/s- 100Mb/s-1Gb/s*	*TP OF*	*Multi-maestro (#limitado)*
LIN	*Bus serie*	*20Kb/s*	*TP*	*1 Maestro(16 Esclavos)*

Tabla 9. Características principales de los protocolos.

[1] TP: Par trenzado; OF: Fibra óptica; OW: Optical Wireless; TSP: Par trenzado blindado.

	Aplicaciones	***Técnicas Tolerancia a fallos aplicadas***
MIL-STD 1553	*Aviación Militar*	*Líneas redundantes* *Bit de paridad* *Impedancia nominal*
ARINC 429	*Aviación comercial*	*Bit de paridad* *RZ*
SpaceWire	*Espacio*	*Las características de tolerancia a fallos no están definidas incluso en las especificaciones*
FlexRay	*Control de alta velocidad en aplicaciones de automoción*	*Líneas redundantes* *De 11 a 24 bits CRC* *Sincronización de reloj* *Detección de data error* *Reconocimiento de trama*
CAN	*Automoción* *Aviónica* *Espacio*	*15 bits CRC* *RZ* *Stuff, Error bit* *Bit Timing* *Form and Acknowledge error* *Reconocimiento de trama*
Ethernet	*LAN Networks*	*No implica la tolerancia a fallos*
LIN	*Automoción*	*Checksum* *Detección del error*

Tabla 10. Aplicaciones y tolerancia a fallos de los protocolos más comunes

La utilización de un protocolo u otro se decide según su velocidad de transferencia de datos y su tipo de comunicación. Para las naves espaciales, aviónica o vehículos automóviles, las comunicaciones de alta velocidad en los sistemas distribuidos no son tan interesantes como para las aplicaciones de protocolos punto a punto, donde un sensor y una unidad de procesamiento de datos se comunican frecuentemente, intercambiando gran cantidad de información. Por otro lado, cuando se especifica el medio físico del enlace de comunicación deben estar considerados el coste del hardware y su peso. Analizando la Tabla 9, para una comunicación que requiera velocidades rápidas, se debe utilizar el bus Ethernet, mientras que el protocolo más rápido de punto a punto es *SpaceWire*. Por otro lado, para aplicaciones de baja velocidad, el protocolo CAN es más barato debido a la disponibilidad de componentes comerciales y la implementación óptica de la comunicación, a la vez que ofrece un alto grado de robustez.

De acuerdo a la información proporcionada por los principales fabricantes de dispositivos de comunicación, cada protocolo incluye características de tolerancia a fallos para entornos críticos de seguridad en distintos grados. Muchos de los protocolos analizados son redundantes, lo cual es muy común en las aplicaciones aeroespaciales o aviónica, e incluyen técnicas redundantes de información (paridad y/o códigos cíclicos de comprobación (CRC)). Algunos protocolos, como ARINC o CAN, tienen implementada la codificación del bit de retorno a cero *RZ* (*return-to-zero*) en la capa física. Los protocolos FlexRay y CAN ofrecen otras características de detección de errores, como el reconocimiento de trama, la sincronización del reloj o bit *timing*, la comparación de bit emitido/recibido (bit de error) y la presencia del bit dominante durante la transmisión.

La tolerancia a fallos incluida en los protocolos de comunicación está directamente relacionada con su campo de aplicación. En este sentido, Ethernet no ofrece ninguna técnica de mitigación del error, igual que las redes LAN no están reguladas para trabajar en ambientes críticos. Por otro lado, FlexRay, CAN bus, etc., están diseñados y desarrollados para los sistemas a bordo en vehículos de motor donde las interferencias son bastante frecuentes, por lo que incorporan numerosas técnicas de robustez.

Hoy en día, algunos de estos protocolos han sido adaptados a una gama más amplia de aplicaciones. Debido al creciente número de aplicaciones con arquitecturas distribuidas, así como la mayor sensibilidad frente a las SEEs de los circuitos modernos, el estudio de cómo las SEEs podrían afectar a la comunicación de los datos es una tarea necesaria. Por esta razón, la importancia de verificar la sensibilidad de protocolos de comunicación ante un SEU real es muy grande.

Hay algunos trabajos de investigación en la literatura científica que analizan la sensibilidad SEU de enlaces de comunicación. En [79], se analiza un controlador de la red Broadcom 57710 que trabaja con protocolos TCI/ IP o UDP bajo la irradiación de neutrones. El experimento de irradiación muestra fallos en la comunicación con la pérdida de datos y la caída del rendimiento.

La conectividad de red en sistemas distribuidos en condiciones críticas debe ser proporcionada por un protocolo de comunicación robusta. En el artículo [80], los autores estudian las interfaces y protocolos utilizados en las aplicaciones espaciales y de la aviónica, centrándose en la selección de los protocolos de ESA. SpaceWire y bus CAN se utilizan en muchas aplicaciones del sector de la aviónica. El protocolo SpaceWire se utiliza en redes de alta velocidad, mientras que el bus CAN se usa cuando se requiere una baja velocidad de datos. ESA utiliza componentes y chips *rad-hard* para integrar ambos protocolos en sus aplicaciones.

Una alternativa a los componentes *rad-hard* es utilizar las tecnologías comerciales y modificar el diseño incluyendo técnicas de mitigación. Este tipo de soluciones será más rentable para alcanzar un alto grado de tolerancia a fallos. En [81] una FPGA de Actel tolerante a la radiación se utiliza para implementar un IP core de SpaceWire. Esta FPGA contiene celdas triplicadas de registros y códigos de Hamming en los bloques de memoria. Sin embargo, esta solución no tiene en cuenta el hecho de que algunos elementos funcionales pueden afectar la sensibilidad ante los SEUs. Endureciendo solamente los elementos más débiles se logrará una mejor optimización en términos de confiabilidad y el coste.

En [82], se presenta un experimento de inyección de fallos en un módulo de bus CAN, con 1.000 *bit-flips* inyectados. Los resultados demuestran la viabilidad de la técnica de inyección de fallos para evaluar los protocolos de la red pero sólo se muestran los resultados relacionados con el rendimiento.

En [84], se presentan las soluciones de tolerancia a fallos para aplicaciones en sistemas basados en el protocolo FlexRay. En este artículo los autores aplican técnicas como TMR en un nodo conectado a dos canales redundantes del FlexRay, utilizando un bloque de control en los nodos, o una distribución de un conjunto de procesos replicados a cualquier número de nodos sin TMR.

La tolerancia a fallos en los sistemas de automoción, donde se utilizan varios protocolos de comunicación interconectados se estudia en [85]. El enfoque propuesto

consiste en el uso de las redes redundantes, pero este artículo no presenta ningún resultado experimental.

La mayoría de los estudios existentes se centran en los protocolos CAN y FlexRay ya que estos protocolos se diseñaron para aplicaciones con requisitos de confiabilidad más altos, que por ejemplo el protocolo LIN. Sin embargo, en la actualidad, hay más y más ECUs (Unidades de Control Electrónico) en los coches lo que provoca que las redes jerárquicas sean necesarias para reducir la carga en el bus principal [86]. El bus LIN puede representar una opción más adecuada para estas subredes, de baja criticidad, debido a su menor coste y a una interfaz más sencilla.

En [85] se analiza un transceptor LIN para detectar los puntos débiles del circuito contra fallos de envejecimiento (fallos permanentes). Este análisis es útil durante la etapa del diseño con el fin de mejorar la capacidad de recuperación. El método propuesto se centra en el análisis de los circuitos con señal mixta.

Los efectos de los errores de software en una red LIN se estudian en el artículo [87]. Los autores proponen un método de evaluación para analizar la fiabilidad en las redes LIN contra fallos *bit-flip* mediante la combinación de dos métodos: un análisis a bajo nivel mediante la inyección de fallos *bit-flip*, y un análisis a nivel de aplicación mediante la simulación de vehículos con modelos de MATLAB/Simulink. Aunque el trabajo es muy interesante y presenta una visión completa, se inyectaron solamente 100 *bits-flips*, que no parece suficiente para conocer las debilidades de tolerancia a fallos de una red LIN.

Como conclusión, podemos decir que en la literatura encontramos numerosos trabajos que han tratado la robustez de los enlaces de comunicación. Sin embargo, en este trabajo de tesis doctoral hemos considerado necesario y conveniente estudiar la robustez frente a los efectos de la radiación ionizante de algunos protocolos representativos de los sectores de la automoción y el sector aeroespacial.

3.4.1. Controller Area Network

El *Controller Area Network* (CAN) es un bus de comunicación serie, desarrollado por Robert Bosch (Alemania) en 1983 y estandarizado por ISO (ISO-11898) en el año 2003. El protocolo CAN está destinado a gestionar la comunicación entre varias unidades de procesamiento con un sistema de árbitro. El CAN fue desarrollado originalmente para la comunicación, con una velocidad media, de los distintos elementos del sistema de propulsión del automóvil (para el control electrónico del motor, frenos, *airbag*, engranaje, sistema de control, etc.), así como para las comunicaciones con los subsistemas de *confort*, de información y de entretenimiento. La comunicación CAN es el protocolo más extendido para las tareas de control y seguridad en los automóviles actuales. Hoy en día, el bus CAN se utiliza para la comunicación entre los nodos que componen los sistemas de control electrónicos en automoción. Casi todos los coches nuevos fabricados en Europa están equipados con una o varias redes CAN, diseñadas para proporcionar una comunicación simple, eficiente y robusta en los elementos internos del vehículo. Además, el bus CAN está cada vez más utilizado en las aplicaciones aeroespaciales.

Las principales características del bus CAN son las siguientes:

- CAN es un bus serie bidireccional, con sistema multi-maestro.
- La velocidad de datos puede ser de 10 Kbps a 1 Mbps. Para una velocidad de datos de 1Mbps la longitud máxima de cable es de 40 metros.

- La capa física soporta el par trenzado diferencial, el cable normal y la fibra óptica.
- Es un protocolo orientado a mensajes con un mecanismo no destructivo de detección de colisiones.
- Tiene varios mecanismos de detección de errores.

Hay dos estados posibles para el bus, el dominante (activo) y el recesivo (inactivo). Si cualquier nodo en el bus está transmitiéndolos datos, el valor dominante manda sobre el recesivo. El nivel recesivo se encuentra en el bus sólo si todos los nodos transmiten el estado recesivo. Este mecanismo, junto con una técnica de prioridad se utiliza con el fin de detectar y resolver posibles colisiones. El principio es el siguiente: si dos nodos comienzan a transmitir simultáneamente, hay al menos un bit en el campo de arbitraje de tramas CAN (que se utiliza para enviar el identificador de trama) en el que se diferencian los nodos. Por ejemplo, si el primer nodo envía este bit como el valor recesivo y el segundo como el dominante, la colisión sería detectada y solo el segundo nodo ocuparía el bus. Esta colisión no es destructiva, ya que el primer nodo detiene la transmisión mientras que el segundo nodo continúa enviando la trama.

Con respecto a los mecanismos de detección de error descritos en la especificación del CAN, cada nodo decide si el mensaje o la trama deben ser desechados o aceptados. Cuando se detecta un error se interrumpe la comunicación y los datos se envían de nuevo. Para el protocolo CAN hay 5 tipos de condiciones de error que se pueden detectar:

- Bit de error. El transmisor también recibe el mensaje enviado y se puede comprobar si los bits están correctos o no.
- *Stuff error*. Hay bits adicionales, sin ninguna información, que se insertan en la trama. El valor y la posición de estos bits dependen del valor de los datos. Cuando se recibe una trama, los bits de relleno tienen que ser eliminados y si se ha producido un error en la transmisión, los receptores pueden detectarlo.
- Error de CRC. Se añade una trama de comprobación de redundancia cíclica, calculada a partir de los datos en el transmisor, al final de las tramas de datos. Los receptores calculan de nuevo el CRC con los datos recibidos y comprueban si el valor obtenido es igual al recibido.
- *Form error*. Se detecta un error cuando los datos de la trama no cumplen con la norma.
- *Acknowledge error*. El transmisor detecta un error cuando después de enviar un mensaje el receptor no responde aceptando el mensaje recibido; en concreto, establece el estado dominante (no hay nodos que reciben el mensaje).

La especificación CAN recoge las propiedades de detección de errores e indica que la probabilidad de error residual total de los mensajes dañados no detectados es inferior a $4{,}7 \cdot 10^{-11}$. Sin embargo, no se especificó cómo se calcula esta probabilidad o qué tipo de errores son considerados. El módulo CAN podría ser sensible a los efectos de los SEUs y los biestables del módulo pueden sufrir los *bit-flips*, que podrían provocar datos incorrectos o errores de control.

3.4.1.1. Implementación del módulo CAN

El estudio se ha realizado sobre un módulo CAN desarrollado por el laboratorio de Observación de la Atmósfera en el Instituto Nacional de Técnica Aeroespacial (INTA). Este IP-*core* cumple con la especificación estándar del CAN. El diseño del bus CAN se ha desarrollado para aplicaciones espaciales como satélites pequeños, donde los requisitos de bajo consumo de energía, el bajo coste, la reutilización y la flexibilidad son obligatorios. En particular, el IP-*core* se utiliza en una arquitectura distribuida de un

ordenador de a bordo del satélite OPTOS [74], donde las principales características del sistema se han detallado en el apartado 2.4.2.5.

Los nodos de la red distribuida del satélite OPTOS se denominan DOT (*Distributed On board computer Terminal*) y son capaces de ejecutar tareas simples. Están conectados a sus correspondientes cargas útiles (*payload*). El subsistema de comunicación a menudo se convierte en una parte del control distribuido.

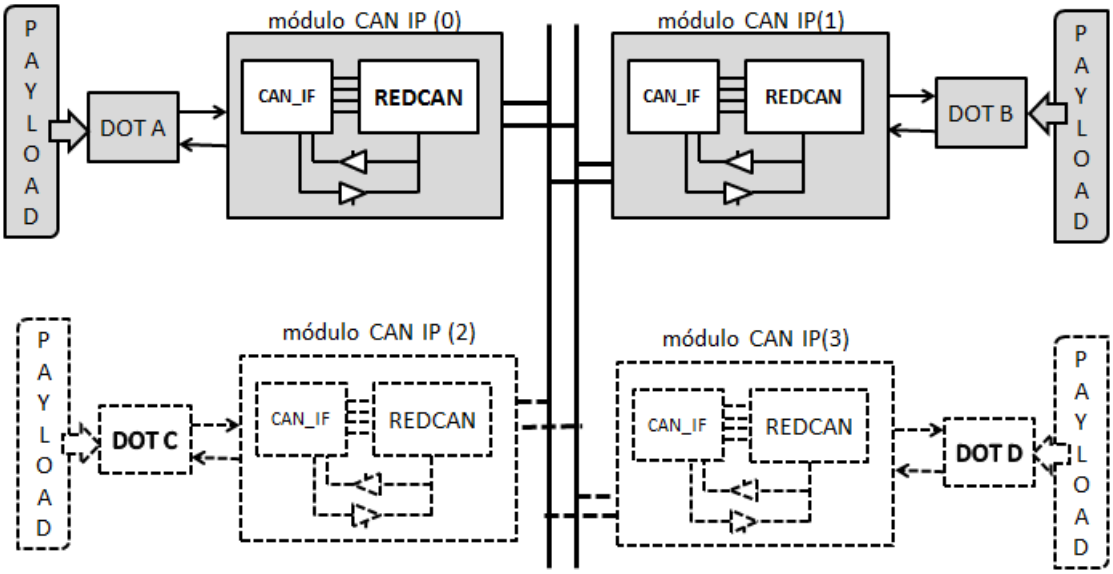


Figura 18. El sistema distribuido con la comunicación por medio de un bus CAN [69]

La Figura 18 muestra los principales bloques del sistema de comunicación utilizado. El sistema consiste de una red compuesta por dos o más nodos interconectados por medio del protocolo CAN implementado. El bloque principal CAN incluye dos bloques internos básicos, RedCan con una funcionalidad definida en la especificación CAN y CAN_IF que es el módulo de interfaz con las DOTs.

3.4.1.2. Análisis de los SEUs en el protocolo CAN

El entorno de inyección de fallos transitorios mediante emulación, al igual que en el caso estudiado anteriormente, corresponde al Sistema de Emulación Autonomo [69].

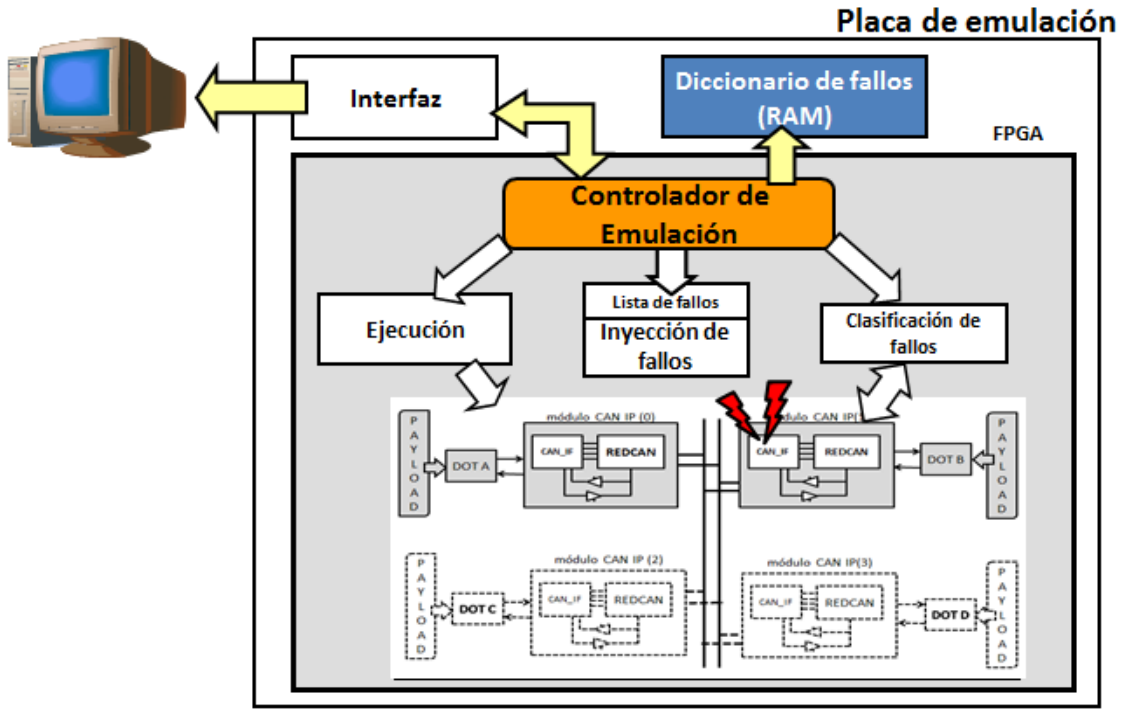


Figura 19. El sistema de emulación de fallos del protocolo CAN [69]

Los fallos inyectados se clasifican por el efecto producido en el comportamiento del circuito. La clasificación utilizada es la siguiente:

- ***Silencioso***. Los fallos se clasifican como un Silencioso cuando el efecto del fallo ha desaparecido por completo y no hay un elemento de memoria que almacene un valor defectuoso.
- ***Detectado***. Un fallo es Detectado cuando algunos de los mecanismos de detección de errores implementado en el módulo CAN son capaces de detectar que un SEU ha afectado al comportamiento del circuito. Entonces, la comunicación se interrumpe y el mensaje se envía de nuevo.
- ***Avería***. Cuando se realiza una transmisión defectuosa, pero el modulo bus CAN no detecta mal comportamiento. Esta categoría incluye tres casos diferentes:
    - Un nodo acepta la recepción de una trama en un instante equivocado, es decir, los datos no han sido enviados o la transmisión no ha terminado todavía.
    - Una trama ha sido enviada y algún nodo afirma la recepción, pero los datos recibidos son erróneos y el mecanismo de detección de errores no los ha detectado.
    - Una trama ha sido enviada y algún nodo afirma la recepción, pero internamente no tiene conocimiento de nuevos datos entrantes.
- ***Latente***. Un fallo es Latente cuando la transferencia de datos se ha realizado de una manera correcta, pero hay algún elemento de memoria con un valor defectuoso. Este tipo de fallos podría producir un error durante las ejecuciones futuras.

El funcionamiento de la red durante la campaña de inyección de fallos incluye diferentes tipos de transmisiones: 66 tramas con 1, 2 o 3 bytes enviados o recibidos, y

16 tramas adicionales que producen colisiones, ya que ambos nodos intentan transmitir al mismo tiempo, es decir el modulo CAN comienza la comunicación con las 82 tramas. Los fallos se inyectaron en cada ciclo de reloj durante la ejecución del funcionamiento de red seleccionado y en todos los flip-flops de uno de los nodos.

3.4.1.3. Resultados experimentales

Se ha inyectado un fallo en cada flip-flop (231 flip-flops) y en cada ciclo de reloj de la carga de trabajo (229.564 ciclos de reloj). El número total de fallos inyectados es 53.029.284.

El circuito se divide en dos grandes bloques, la Interfaz CAN (IF) y la RedCan (CORE). La interfaz conecta el core CAN con un circuito del usuario. El circuito de interfaz se ha dividido en los flip-flops de datos (IF_DATA) y los flip-flops de control (IF_CTRL).

El circuito (CORE) se divide en los bloques del control (CORE_CTRL), del cálculo y comprobación del CRC (CORE_CRC), del manejo de errores (CORE_ERR), de sincronización (CORE_SYNC), del circuito de recepción (CORE_RX) y transmisión (CORE_TX) y de CORE_STAFF.

La Tabla 11 muestra la clasificación de los fallos agrupados por los bloques. En esta tabla, los fallos han sido clasificados como Averías, Latentes, Detectados y Silenciosos. Se muestra el número total de flip-flops y la distribución de los fallos para cada uno de los bloques. La última línea muestra la clasificación del fallo global.

Bloque	***#FF***	***Averías***	***Latentes***	***Detectados***	***Silenciosos***
IF_DATA	39	3.712.096	44.207	0	5.196.693
IF_CTRL	25	839.359	1.844.181	22.608	3.032.952
CORE_CTRL	10	581.612	240.686	73.222	1.400.120
CORE_CRC	15	487.528	3.038	662,042	2.290.852
CORE_ERR	51	1.059.624	1.774.528	2.699.964	6.173.648
CORE_SYNC	29	969.511	26.497	894.240	4.767.108
CORE_RX	43	381.764	63.414	270.155	9.155.919
CORE_TX	13	0	19.049	0	2.965.283
CORE_STUFF	6	0	3.918	0	1.373.466
Total	***231***	***8.031.494***	***4.019.518***	***4.622.231***	***36.356.041***
		15,15%	**7,58%**	**8.72%**	**68,56%**

Tabla 11. Clasificación de fallos agrupados por bloques

Como resultado global, alrededor del 68,56% de los fallos son Silenciosos y 7,58% son Latentes. Los fallos Latentes aparecen porque hay partes del circuito que no se utilizan, sobre todo las que están relacionadas con la lógica de error (error CRC, etc.). Hay varios tipos de errores que no han sido evaluados por la carga de trabajo. Debe tenerse en cuenta que no se han probado los errores de datos (errores en el canal de comunicaciones, como el ruido), y hay una lógica destinada a tratar estos errores. Las tasas de fallos silenciosos y detectados (cerca de 77%) indican que el circuito en general es bastante robusto.

Los resultados obtenidos han demostrado que más del 15% de los fallos inyectados en el protocolo CAN producirán una Avería, es decir, el bus va a llevar a cabo transmisión defectuosa pero el modulo no va a poder detectar ningún mal comportamiento. Alrededor de un 20% de los elementos de la memoria en el módulo CAN está provocando las Averías. En la Figura 20, se muestra la clasificación de los fallos para cada FFs del circuito. Estos están agrupados por bloques funcionales. Como se puede ver, los bloques más sensibles son los de interfaz, los de data y los de error. El

endurecimiento de estos elementos implicaría una disminución significativa en la tasa de Averías, del 15,15% a 3,98%, Tabla 12.

Número de FFs endurecidos	Tasa de errores
0	15,15%
10	11,69%
20	9,16%
30	7,19%
40	5,46%
50	3,98%
60	2,79%
70	1,78%
80	0,97%
90	0,42%

Tabla 12. Porcentaje de *flip-flops* endurecidos y su ratio de averías correspondiente.

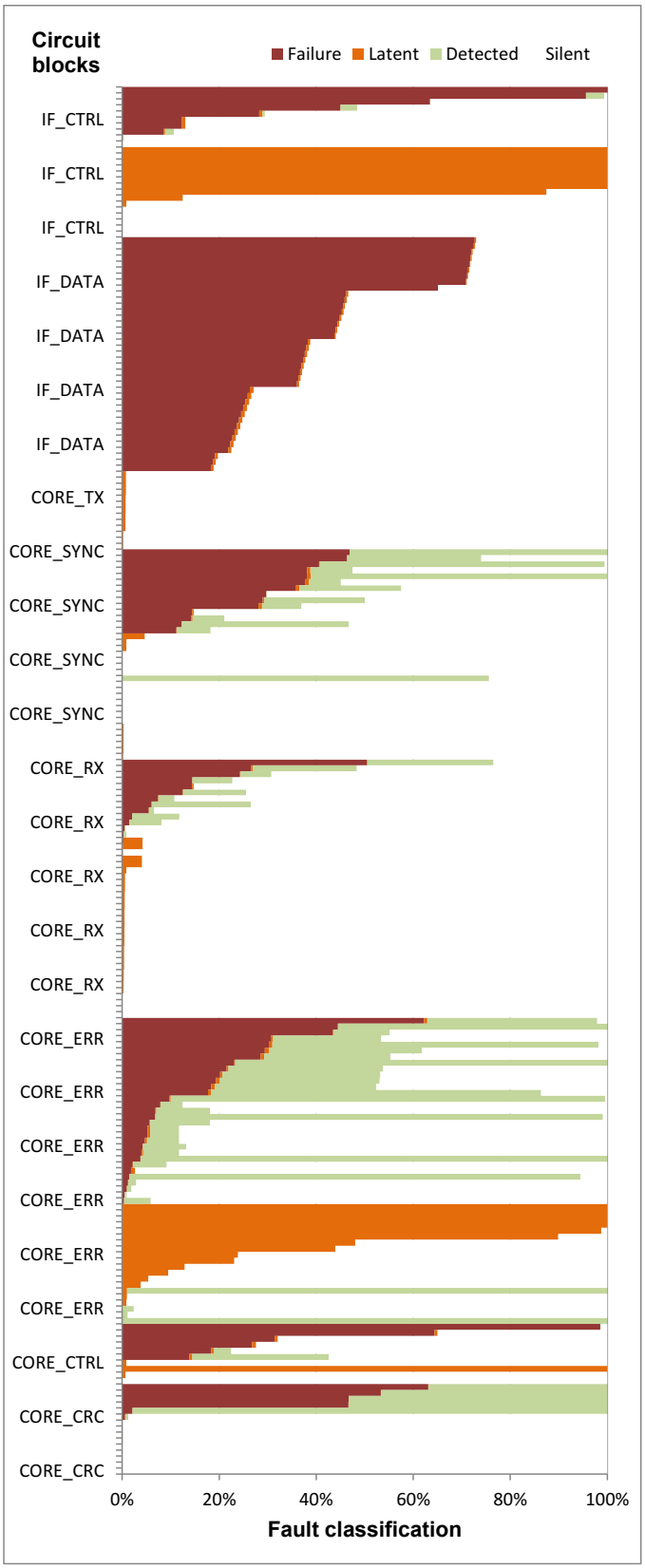


Figura 20. Clasificación de los fallos en el CAN bus en función de los bloques [69].

3.4.2. Protocolo LIN bus

Local Interconnect Network es un bus de comunicación serie, utilizado mayoritariamente para sistemas electrónicos distribuidos y más concretamente para las tareas del confort de los vehículos. Fue especificado y desarrollado por un consorcio compuesto por cinco fabricantes de automóviles (Audi, BMW, Volkswagen, Volvo, Daimler-Chrysler), un proveedor de semiconductores (Motorola) y un proveedor de herramientas CAD (*Volcano Communications*). Hay varias versiones del estándar LIN. La versión 1.3 es la primera en la cual se especifica la funcionalidad del protocolo. Las versiones 2.0 y 2.1 añaden más especificaciones de mensajes y servicios pero son completamente compatibles al nivel de byte con la versión 1.3 del LIN. La especificación 1.3 de LIN es utilizada mayoritariamente por algunos fabricantes de equipos franceses y japoneses, el resto de los fabricantes de vehículos europeos mayoritariamente están utilizando la especificación LIN 2.0. LIN 2.1 es una versión modificada y mejorada de esta especificación y consta de 8 puntos:

- Especificación de la capa física.
- Especificación del protocolo.
- Especificación de la capa de transporte.
- Configuración del nodo e identidad.
- Especificación del diagnóstico.
- Especificación de la interfaz de programación de las aplicaciones.
- Especificación de la capacidad del lenguaje del nodo.
- Configuración de la especificación del lenguaje.

La Sociedad de Ingenieros de Automoción (*Society of Automotive Engineers*) J2602, recomienda el uso del protocolo LIN para las aplicaciones de automoción en América del Norte.

Las principales características de protocolo LIN son:

- La comunicación del protocolo es serie.
- El bus es de un único cable que trabaja a 12V, con dos tipos de niveles dominante y recesivo que cumplen con la normativa ISO-9141 y el estándar NRZ.
- La velocidad de transmisión es baja (20 kbps).
- El protocolo LIN tiene un único maestro y varios esclavos (hasta 16 nodos). El protocolo LIN no utiliza arbitraje tipo bus. Un solo maestro es responsable de inicializar todas las transferencias. Todos los esclavos pueden leer el mensaje enviado por el maestro, pero solo llevaran a cabo la acción enviada en el mensaje aquellos a los que el maestro asigne a través de una dirección.
- Longitud variable de trama de datos (2, 4 y 8 bytes): existe un byte dentro del mensaje que se encarga de determinar la cantidad de datos que se van a enviar, permitiendo de esta manera disponer de un sistema flexible, y con la cantidad de datos óptima.
- La longitud máxima del bus es de 40 metros.

- De acuerdo a las especificaciones el comportamiento del bus es predecible ante Interferencias Electromagnéticas, *EMIs*.
- Presenta la capacidad de detectar el nodo fallido. El maestro tiene la responsabilidad de solicitar información del estado de los esclavos y comprobar que todos los nodos funcionan correctamente.

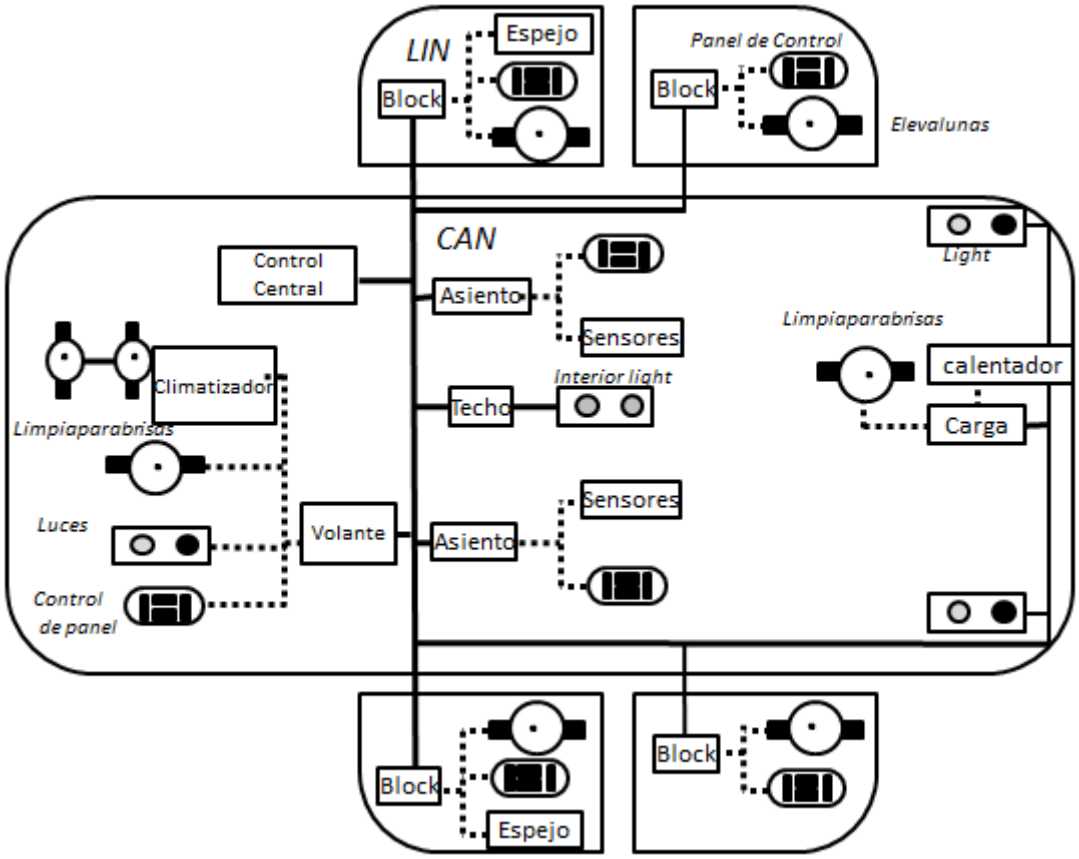


Figura 21. Aplicación típica del LIN bus en el coche actual [75].

El bus LIN ha sido desarrollado para mejorar y ampliar el uso de los dispositivos electrónicos en el coche, haciendo el vehículo más cómodo y seguro con un coste de implementación relativamente bajo y con una escalabilidad del sistema elevada. La posibilidad de añadir nueva funcionalidad a un nodo esclavo sin repercutir en los nodos esclavos restantes, mejora enormemente la flexibilidad y escalabilidad de los sistemas distribuidos.

Se pueden considerar diferentes opciones para el desarrollo de un nodo LIN. En primer lugar, la solución más barata y la más fácil de implementar es el desarrollo del código software del nodo LIN en un microcontrolador comercial, tal como es PIC® de Microchip. Otra solución consiste en el desarrollo personalizado que es posible mediante el uso de un lenguaje de descripción de hardware y con el posterior prototipado en una FPGA, o en un ASIC si se esperan grandes cantidades a utilizar. Esta solución debe ser analizada cuidadosamente, especialmente si se elige la tecnología SRAM y si la aplicación final va a trabajar en un ambiente crítico.

La principal ventaja del protocolo LIN es su bajo coste, pero es una solución menos robusta que, por ejemplo, el protocolo CAN bus. Esta es la razón para estudiar el uso de este protocolo en las aplicaciones no tan críticas o para tareas de procesamiento

de datos no críticos (el bus LIN presenta un ancho de banda bajo y una baja tasa de bits).

Actualmente, el bus LIN se utiliza en los sistemas distribuidos de los automóviles: en sensores y actuadores de los elementos del automóvil tales como las puertas, el volante, los asientos, la climatización, los diferentes motores, las luces, el sensor de lluvia, los limpiaparabrisas inteligentes, los espejos retrovisores, etc. Sin embargo, aunque la tasa de fallos podría ser baja, de acuerdo con las normas vigentes, un error de software en cualquiera de estos sistemas podría arriesgar la vida humana durante el viaje. Dicho esto el endurecimiento de estos sistemas no críticos puede resultar interesante si es eficaz y económico.

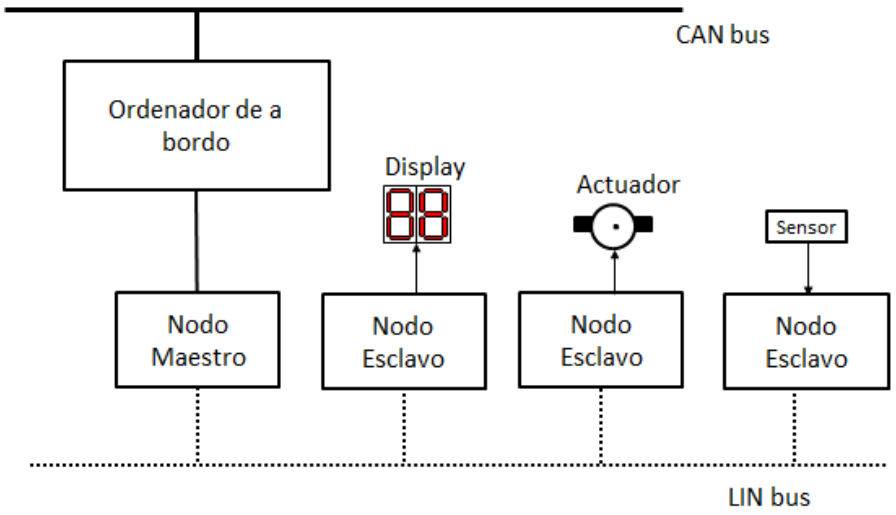


Figura 22. El esquema de una red distribuida conectada por el bus LIN [76].

Una red de sensores y actuadores inteligentes conectada a través del protocolo LIN siempre contiene un nodo maestro que está a cargo del interfaz con un microprocesador de a bordo (PIC, 8051, etc.), que normalmente se conecta con redes superiores de otro sistema distribuido (por ejemplo protocolo CAN en los coches). En la Tabla 13 se muestra una comparativa de las características del bus CAN y el bus LIN. A pesar de que el bus LIN tiene una arquitectura más sencilla y menos potente comparando con el bus CAN, el bus LIN tiene una estructura fácil de implementar y puede servir como una base para cualquier otro protocolo de comunicación

Características	**LIN**	**CAN**
Control de acceso	Maestro Único	Múltiples maestros
Velocidad del bus	2,4, 9,6 y 19,2 Kbps	62.5 Kbps – 1 Mbps
Rutado del mensaje	6 bit de identificador	11/29 bits de identificador
Tamaño de la red	De 2 a 16 nodos	De 4 a 20 nodos
Numero de bytes de datos por marco enviado	De 0 a 8	De 2 a 8
Tiempo de transmisión por 4 bytes de datos	6 ms a 19,2Kbps	0,8 ms a 125Kbps
Detección de error	*Checksum* de 8-bit	CRC de15 bits
Capa física	Un solo hilo, 40V	Par trenzado blindado, 5V
Cuarzo/Resonador cerámico	Solo para el maestro	Para todos los nodos

Tabla 13. La comparativa de las características principales LIN y CAN

3.4.2.1. El funcionamiento del nodo de Control LIN (LIN Controller)

El modulo controlador LIN se encarga de toda la serialización, de los tiempos a nivel de bits, de la formación de las tramas, de la generación y la validación del checksum, de la generación y la validación de los bits de paridad y garantiza y valida las tramas del protocolo LIN [75]. La interfaz física se realiza por un transceptor LIN externo, como se muestra en la Figura 23.

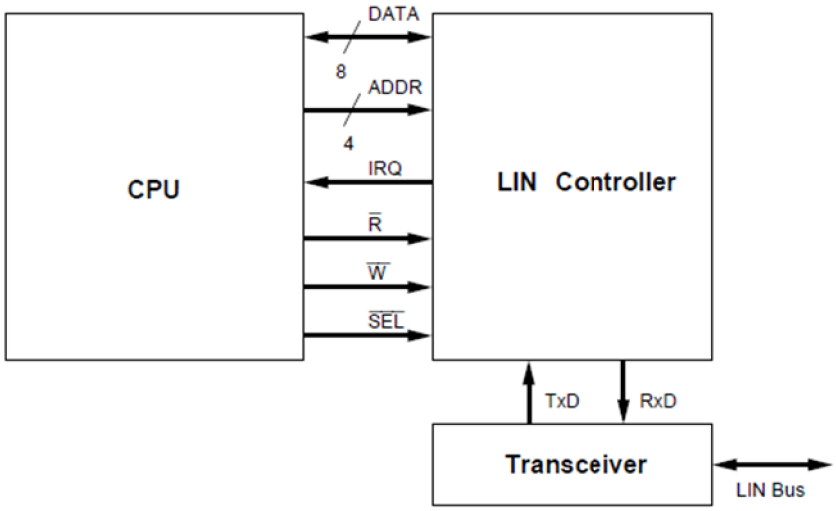


Figura 23. La interfaz del controlador LIN [76].

Una trama del protocolo LIN se compone de una cabecera y una respuesta. La cabecera se envía siempre por el nodo maestro LIN, mientras que la respuesta es enviada por cualquiera de los nodos. Para enviar o recibir los datos por el protocolo la CPU principal del sistema interactúa con el controlador LIN fijando la dirección del controlador (ADDR), a través del bus de datos de un ancho de 8 bits. El controlador puede interrumpir la CPU cuando aparezca un error (INT), o una recepción o transmisión de datos correcta. La CPU genera todos los datos necesarios para el protocolo LIN por código software, y está conectada a la red LIN a través de un transceptor LIN. Para asegurar una tasa de estabilidad de transmisión dentro de una trama LIN, se utiliza una parte del mensaje de trama (SYNC) dentro de la cabecera. El protocolo LIN implementa algunos esquemas de detección de errores, como la comprobación de paridad de errores (*parity-error checking*) y la comprobación de suma de errores (*checksum-error checking)*. También los errores en el procesamiento de la trama son detectados y reportados. Cuando se detecta algún error, el maestro informa a la CPU, con el fin de repetir el comando o realizar otra acción.

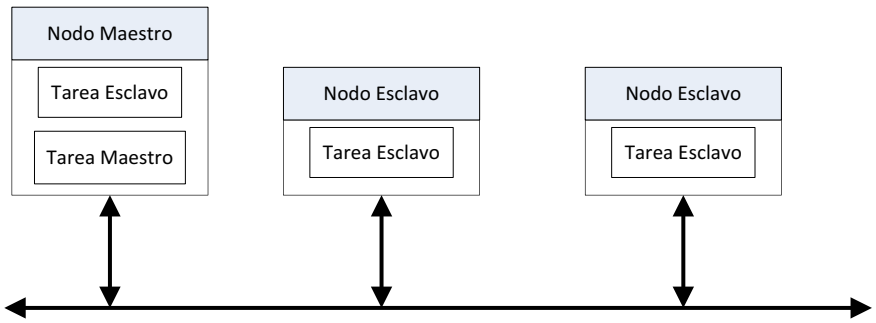


Figura 24. Bus LIN.

Toda operación en un sistema con la comunicación a través del bus LIN va a disponer de tres componentes principales y de las relaciones entre estos. Estos elementos son el nodo maestro, el nodo esclavo y la trama.

En el protocolo LIN existen dos tipos de nodos, el nodo maestro (que incluye un nodo esclavo asociado a él) y los nodos esclavos. El nodo maestro siempre decide cuándo y qué trama se transferirá en el bus, indicando a través de la trama que esclavo se activará y que tarea debe hacer.

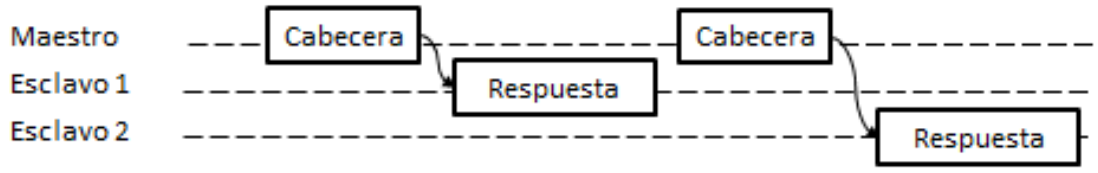


Figura 25. La comunicación entre el maestro y los esclavos en el protocolo LIN [75].

De acuerdo a la especificación del protocolo LIN [75] la información se transmite a través de las tramas (*frame*). Una trama consta de una cabecera (*Header*) y una respuesta (*Response*). La cabecera consiste en 14 bits de interrupción (*break*), un byte de sincronización (*sync*) y de un byte de identificador de trama (*identifier*). Entre los bytes de sincronización de identificador hay unos bits delimitadores.

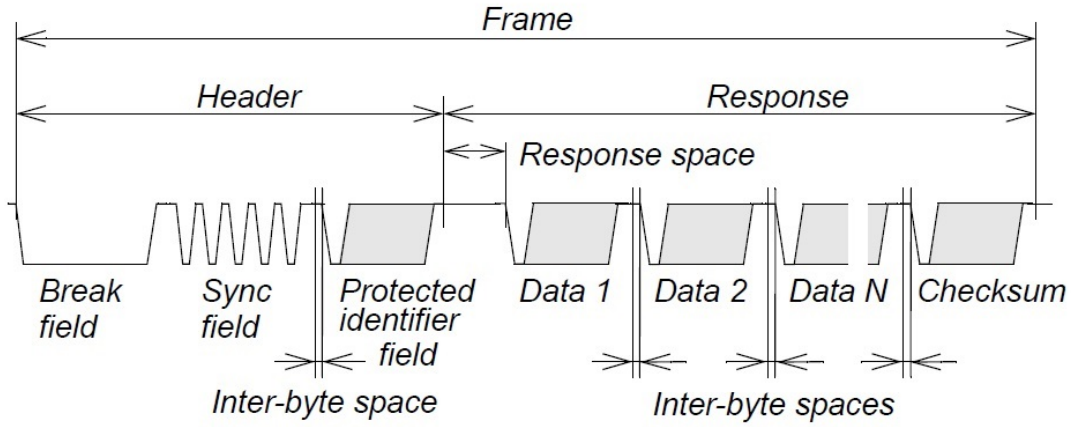


Figura 26. La estructura de un mensaje LIN[75].

La longitud de la trama de la cabecera consiste en un total de 34 bits.

$$T_{header\ nominal} = 34 \cdot T_{bit}$$

La respuesta consiste en los bits de datos (*date field*) y dos bits de comprobación (*checksum*). Los esclavos del protocolo pueden enviar un máximo de 8 bytes de datos.

$$T_{response\ nominal} = 10 \cdot (N_{data} + 1) \cdot T_{bit}$$

Break

Este campo nos indica el comienzo de una nueva trama. Siempre va a ser generado por el nodo maestro, y deberá tener 13 bits de valor dominante, y acabar con delimitador de rotura (*break delimiter*), como se muestra en la Figura 27.

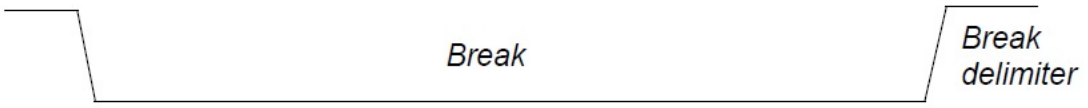


Figura 27. La estructura del *break* de un mensaje LIN [75].

Sync byte

El byte de sincronización al igual que el *break* se encarga de indicar al nodo esclavo de que se va a enviar una nueva trama, tiene un valor de 0x55 en hexadecimal, y siempre tiene la misma estructura.

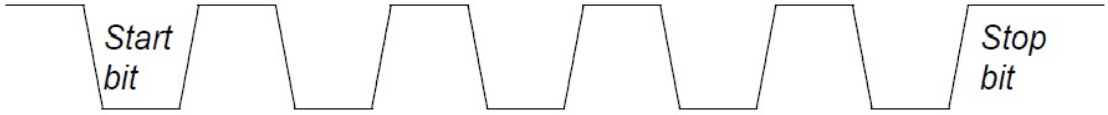


Figura 28. La estructura del byte de sincronismo del mensaje LIN [75].

Un esclavo siempre será capaz de detectar la secuencia de la sincronización, incluso si está esperando otro tipo de datos. Siempre que aparezca el byte de la sincronización, los datos en curso serán abortados y comenzará el procesamiento de envío de la nueva trama.

***Identifier Field* (Identificador ID)**

El identificador del mensaje está formado por 8 bits, con una determinada estructura y se divide en dos partes: el identificador de trama que corresponde a los bits del 0 al 5, y los bits de paridad 6 y 7.

IDENTIFICADOR DE UN MENSAJE LIN							
Bit 7 P1(MSB)	Bit 6 P0	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Bits de paridad	***(identificador de la trama) 6 bits (64msg)***						

Tabla 14. La estructura del identificador de un mensaje LIN.

El identificador está formado por 6 bits, que se van a dividir en tres categorías de acuerdo al valor que tomen:

- Los valores de 0 a 59 (0x3B) se utilizan para las tramas que transportan los datos,
- 60 (0x3C) y 61 (0x3D) se utilizan para transportar los datos de diagnóstico y configuración,
- 62 (0x3E) y 63 (0x3F) se reservan para futuras mejoras del protocolo.

El identificador contiene la información relativa al nodo receptor y el número de bytes que se esperan recibir.

Los bits de paridad se calculan sobre los bits del identificador de trama como se muestra en las siguientes ecuaciones:

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$
$$P1 = NOT\ (ID1 \oplus ID3 \oplus ID4 \oplus ID5)$$

Dato

Una trama puede contener entre uno y ocho bytes de datos. Dentro de los bytes, el primer bit enviado es el LSB y el último MSB.

Dato							
Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)

Tabla 15. Los bits del dato de la comunicación LIN.

Dos de los bits de la trama, opcionalmente, pueden indicar el tamaño de la misma.

Checksum

El *checksum* es una comprobación de la trama mediante la realización de una suma. El *checksum* contiene la suma invertida de los ochos bits de todos los bytes de datos (el *checksum* clásico es utilizado en el nodo maestro para solicitar la trama, en la respuesta del esclavo y en la comunicación entre esclavos del protocolo) o todos los bytes de datos y el *ID* (el *checksum* mejorado utilizado en la versión LIN 2.1). El byte de verificación, que se transmite a través del bus, es el complemento bit a bit de la suma de comprobación. Por lo tanto, para calcular el *checksum* correcto, una trama que se acabe con "00" es la correcta.

3.4.2.2. Implementación hardware

El circuito implementado toma como punto de partida la nota de aplicación [75] de Xilinx. El diseño se ha modificado para hacerlo totalmente síncrono, configurable y genérico. Se añade una interfaz para la comunicación con el bus APB para una comunicación entre el maestro y el microprocesador. Cada nodo LIN se divide en ocho bloques, como se muestra en la Figura 29, [76]. El bloque transmisor y el receptor se encargan de recibir y transmitir los datos, mientras *Parity_Generator* y *Checksum_Generator* se encargan de la detección de errores en los datos recibidos. El transmisor y el receptor están implementados con dos registros de desplazamiento, mientras que los datos recibidos son preprocesados en el bloque *Majority_Sampler* que detecta los bits de acuerdo al tiempo estándar. La parte de control del circuito está compuesta por los bloques *Configuration_Registers* y un bloque de *Control*, que se compone de un conjunto de máquinas de estado. La recepción de la trama y la composición de la misma se controla por dos (en caso del esclavo) o por tres (en caso del maestro) máquinas de estados, que se encuentran en el bloque de control. Estas máquinas de estados también se encargan de la detección y el informe de errores, así como de la decodificación de cabecera. En el siguiente diagrama de bloques se puede ver la interrelación entre los distintos elementos dentro del controlador LIN.

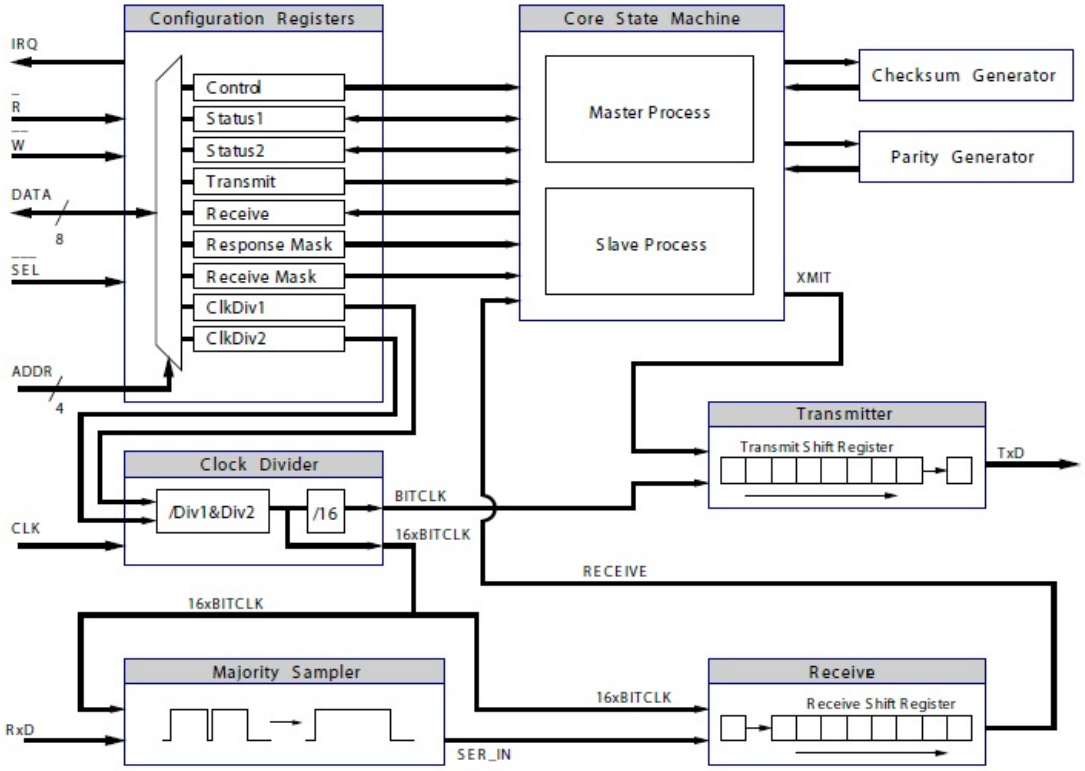


Figura 29. El diagrama de bloque completo del protocolo LIN [76]

El bloque de configuración de los registros se encarga de la gestión de la interfaz con la CPU, y realiza la inicialización y el ajuste de los registros del estado y de configuración. La interfaz con la CPU actúa como un conjunto de registros direccionables de un ancho de 8 bits, algunos de los cuales se puede manejar por medio de la CPU, y otros que sólo pueden ser accedidos por el propio controlador LIN. Los registros, sus funciones y los valores iniciales de los registros se enumeran en la Tabla 16.

CONTROL	0x0	Registro de la configuración de los nodos
STATUS1	0x6	Registro de estado de los mensajes
STATUS2	0x0	Maneja todos los tipos de errores posibles del protocolo.
TRANSMIT	0x0	Registro de transmisión de datos
RECEIVE	0x0	Registro de recepción de datos
IDMASK	0x0	Slave ID MASK
IDFILTER	0x0	Slave ID FILTER
CLKDIV1	0x2	Divisor de reloj 1
CLKDIV2	0x0	Divisor de reloj 2

Tabla 16. Registros de configuración con los valores de inicialización.

El registro de configuración de los nodos CONTROL configura el nodo como maestro o esclavo (MASL), inicializa la frecuencia del funcionamiento del protocolo (ABAU), maneja la interrupción del error (IERR) y controla la interrupción de la transmisión (ITX) y recepción (IRX) del dato.

El registro STATUS1 verifica el estado de la transmisión (TDRE) y de la recepción (RDRE), avisa en caso de que se haya producido un error en el protocolo (ERROR) y verifica en qué estado se encuentra la transmisión o la recepción (IDLE).

Bits	Función	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
0000	CONTROL	IRX	ITX	IERR				ABAU	MASL
0010	STATUS1					ERROR	IDLE	TDRE	RDRE
0011	STATUS2		ESH	OVRL	XMIT	FRAM	PAR	CKS	ORUN
0100	TRANSMIT								
0101	RECEIVE								
0110	IDMASK			Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0111	IDFILTER			Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1000	CLKDIV1	Bit7	Bit6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1001	CLKDIV2					Bit 11	Bit 10	Bit 9	Bit8

Tabla 17. Los valores de los bits de los registros de configuración del protocolo LIN.

El protocolo LIN comprueba la ocurrencia de distintos tipos de errores y escribe la información relativa a estos errores en el registro STATUS2:

- Error de trama corta (shortframe-ESH).
- Error de no respuesta del Esclavo (noslaveresp-OVRL).
- Error en la transmisión (e_xmit -XMIT).
- Error de la trama (framing-FRAM).
- Error de la paridad (parity-PAR).
- Error del checksum (cksum -CKS).
- Error del overrunning (ORUN).

La máquina de estados controla todas las operaciones de sincronización, activa varios registros de estado e interrumpe la CPU cuando sea necesario. La máquina de estados está compuesta por una máquina de estado maestra y otra esclava. También contiene un contador que se encarga de sincronizar todas las operaciones, un detector del *break* que detecta cuando se inicializa la trama, un contador de bytes, que cuenta el número de bytes que se ha indicado anteriormente en el identificador.

El maestro se encarga de controlar el arbitraje del bus, debe iniciar la respuesta de cualquier esclavo y debe arbitrar e iniciar cualquier comunicación esclavo-esclavo. Funcionalmente, la máquina maestra envía un *break*, un *sync*, y un primer byte del identificador del esclavo. La máquina maestra va al estado de reposo, se activa la máquina esclava y se espera hasta que el esclavo envíe una respuesta.

La tarea esclava es responsable del envío del dato. Si el esclavo ha sido direccionado por el maestro (indicándolo en la parte de la trama correspondiente), este responderá o llevará a cabo la acción dependiendo del contenido del mensaje.

El controlador LIN tiene una máscara ID configurable y un filtro, que son comparados y si no concuerdan la trama recibida será completamente ignorada, hasta la llegada del próximo break. Los filtros son una manera de reducir el trabajo en los nodos esclavos. Estos solo deben atender a un número pequeño de identificadores. En cambio, el nodo maestro no necesita utilizar ninguna máscara o filtro.

El receptor ha sido implementado con un registro de desplazamiento, y dispone de una salida en paralelo y una señal de inicialización asíncrona *(Reset*). El *reset* está conectado a la máquina de estados que lo pone a cero si fuese necesario al receptor la recepción de un nuevo byte. El receptor mostrará un mensaje de error si encontramos

algún defecto en la trama, y sacará una salida correcta cuando la recepción haya sido completa. El transmisor se implementa con otro registro de desplazamiento y tiene una entrada en paralelo, comprueba si se han desplazado todos los bits y proporciona la salida del error si el último bit de la transmisión es diferente del bit recibido, indicando una avería en el bus o un error transitorio.

El divisor de reloj divide el reloj del sistema con un divisor de 16 bits. El MSB de este divisor corresponde al registro CLKDIV1, y el LSB corresponde al registro CLKDIV2. A continuación, este reloj interno se divide por 16, generando un reloj interno *bitclk*, que se usa para la mayoría de las operaciones de temporización. Para calcular el divisor del reloj se utiliza la siguiente fórmula:

$$Divisor = \frac{CLK}{16 \cdot bitrate} = CLKDIV1\&CLKDIV2$$

La especificación LIN requiere que los nodos esclavos tengan la capacidad de ajustar sus relojes locales utilizando la trama de sincronización enviada por el maestro. El módulo implementado trabaja con un reloj de 10 MHz con una tasa de datos de 9.600 bps.

El diseño ha sido prototipado en una CPLD CoolRunner-II. Los resultados son similares a los obtenidos por Xilinx en su diseño original, como se muestra en la Tabla 19.

	Macro celdas	Lógica	Registros	Pines	Function Blocks Inputs
TOTAL	256	896	256	118	640
XAPP432	199	696	168	20	505
UC3M	245	686	169	20	550

Tabla 18. La ocupación del área del circuito bus LIN implementado por Xilinx y por UC3M.

3.4.2.3. Análisis de sensibilidad ante los SEUs.

El análisis de la sensibilidad ante los SEUs del módulo controlador del bus LIN se ha realizado con el fin de conocer los efectos de los fallos transitorios que afectan a la transmisión / recepción de datos y con el fin de detectar las áreas más sensibles en el controlador LIN. Endureciendo de forma selectiva las áreas débiles, se puede realizar una mejora en la robustez del módulo controlador. Una vez que el módulo esté endurecido, un análisis más detallado debe de llevarse al cabo en cuanto a la detección de los fallos permanentes que afecten al circuito durante el funcionamiento normal.

El análisis de los SEUs se realiza mediante una campaña de inyección de fallos. El sistema bajo prueba (red LIN) consiste en varios nodos LIN interconectados entre sí. En particular, se han implementado un controlador maestro y un controlador esclavo, aunque se podrían conectar hasta un máximo de 16 esclavos. La campaña de inyección de fallos se ha realizado mediante el uso del sistema de Emulación Autónoma. Por lo tanto, todas las tareas de inyección fueron realizadas por medio de *hardware* como se puede apreciar en la Figura 30.

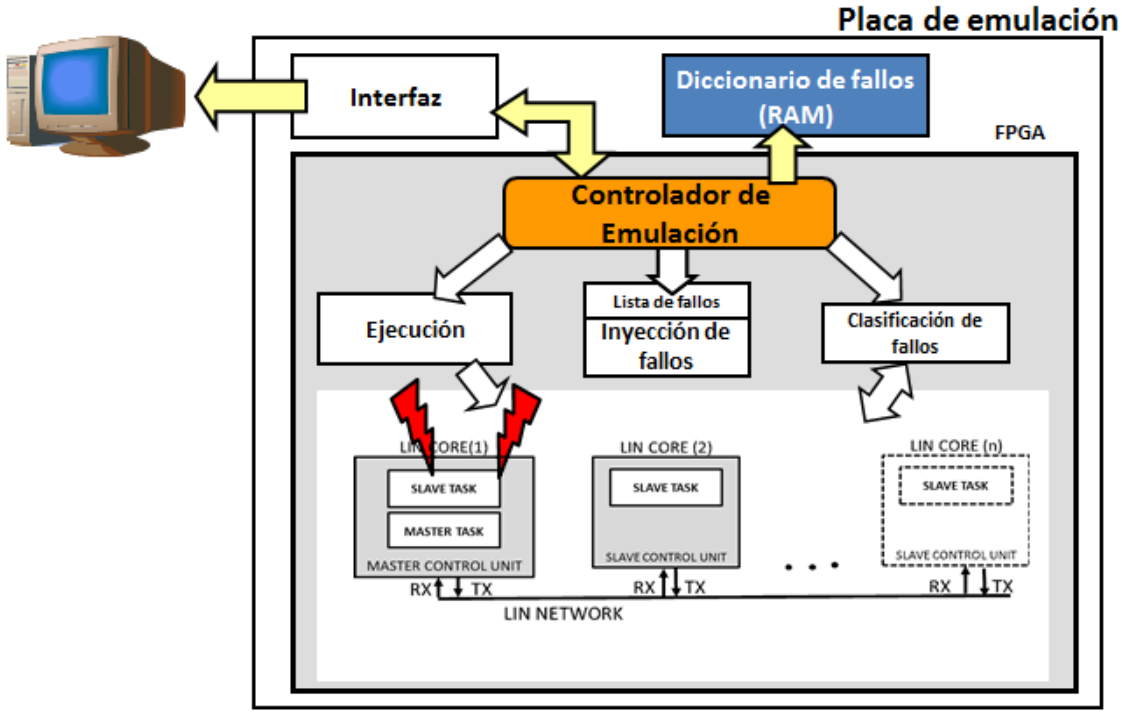


Figura 30. El sistema de emulación del LIN *network* [96].

La red de los nodos LIN ha sido prototipada en una FPGA Virtex5 de Xilinx. Los fallos inyectados se han clasificado acorde al efecto producido en el comportamiento del sistema:

1. El efecto de los fallos se propaga a las salidas:
   - Un fallo se clasifica como Detectado cuando alguno de los mecanismos de detección de errores implementado en el módulo LIN es capaz de detectar que un SEU ha afectado al comportamiento del circuito.
   - Un fallo se clasifica como Avería cuando se realiza una transmisión defectuosa pero el controlador LIN no detecta ningún comportamiento anómalo.
2. El efecto del fallo no se propaga a las salidas:
   - Un fallo se clasifica como Silencioso si su efecto ha desaparecido por completo dentro del circuito.
   - Un fallo se clasifica como Latente si sus efectos han quedado en algunos elementos de la memoria del circuito.

En el banco de pruebas de la red LIN durante la campaña de inyección de fallos ha sido seleccionada una carga de trabajo que incluía once diferentes pruebas. En primer lugar, había cuatro pruebas básicas de transmisión de los datos desde el controlador 1 al controlador 2. Las tres primeras pruebas enviaban 2, 4 y 8 bytes, respectivamente, mientras que la cuarta prueba enviaba un dato de identificación y el controlador 2 debía de responder. Las pruebas básicas verificaban que los dos nodos se comunicaban correctamente. Se han añadido también otros *tests*, como el *test* de error de paridad, la detección de errores de suma de comprobación y el error de la trama que requería forzar el bus directamente. La forma general de una prueba consistía en escribir en un registro de configuración, esperar hasta que se produjera una interrupción ya sea por la recepción del dato o por un error, y a continuación se actuaba ante esta interrupción por

medio de la lectura de un byte recibido o mediante el control de los diversos registros de estado.

3.4.2.4. Resultados experimentales

La campaña de inyección de fallos se ha realizado en los dos controladores LIN, uno que actuaba como maestro y el otro como esclavo. Se incluyen solo los resultados del controlador maestro ya que los resultados del esclavo no añaden ninguna información adicional.

Los fallos han sido inyectados en cada ciclo de reloj (de los 37.381 ciclos de reloj) durante la ejecución del banco de pruebas y en todos los flip-flop (205 flip-flops). El número total de fallos inyectados fue de 7.663.105. El sistema de Emulación Autónoma puede alcanzar velocidades de inyección de fallos de hasta millones de fallos por segundo. Por lo tanto, el tiempo de ejecución no es un problema para emular un sistema en el que se inyectan cientos de millones de fallos, ya que toda la campaña dura varios minutos.

La Tabla 19 muestra la clasificación de los fallos correspondientes a todos los bloques del circuito. Estos bloques corresponden a los registros de configuración, los bloques de recepción, el divisor de frecuencia, el bloque de control, el generador de checksum y el bloque transmisor.

Como resultado global, el 28% de los fallos inyectados causan como resultado una comunicación errónea (Avería), el 51% no tiene ningún efecto (Silencioso), el 18% son Detectados por los mecanismos de comprobación del circuito, y un 3% corresponde a los fallos Latentes en el circuito al final de la carga de trabajo.

	#FF	Avería	Latente	Silencioso	Detectado
CONF_REG	56	1.084.510	67.634	595.546	345.646
RECEIVER	23	47.623	13.576	774.221	24.343
MAJ_SAMPLER	4	8.339	663	131.703	8.819
DIVIDER	17	200.826	57.381	18.874	358.396
CHECKSUM	9	23.297	11.961	301.171	0
CTRL_CHECKSUM	2	0	2	74.760	0
CTRL_COUNTER	22	46.737	12.109	668.495	95.041
CTRL_ERROR	9	133.413	9	43.296	159.711
CTRL_FRAME	4	12.330	1.716	99.889	35.589
CTRL_MASTER	9	88.321	14.783	173.820	59.505
CTRL_SLAVE	23	264.538	54.207	303.278	237.740
CTRL_RECEIVE	9	143.144	4.240	186.477	2.568
CTRL_SER_IN_LAST	1	0	18	37.363	0
CTRL_STATUS	3	19.794	2.885	65.315	24.149
TRANSMITTER	14	78.866	9.559	410.700	24.209
Total	205	2.151.738	250.743	3.884.908	1.375.716
		28,08%	3,27%	50,70%	17,95%

Tabla 19. Clasificación de los fallos inyectados en el controlador LIN

La parte más crítica del circuito corresponde al bloque de los registros de configuración, seguido por determinadas partes del bloque de control y el divisor de frecuencia.

Probablemente no todos los fallos detectados en este nivel serían propagados a los niveles superiores. Pero esta prueba permite realizar una identificación de los elementos críticos del circuito que provocarían fallos que sí que podrían realmente propagarse al nivel de sistema y causar un fallo en el mismo. El endurecimiento de estos elementos críticos a nivel de circuito resulta ser más económico y con una complejidad menor frente al que se realizaría en los niveles superiores.

La Tabla 20 muestra una comparación entre la inyección de fallos en los protocolos CAN y LIN.

	#FF	Avería	Latente	Silencioso	Detectado	Fallos
CAN	*231*	*12.08%*	*7.81%*	*64.66%*	*15.46%*	*53.0M*
LIN	*205*	*28.08%*	*3.27%*	*50.70%*	*17.95%*	*7.6M*

Tabla 20. La comparativa de las clasificaciones de fallos entre protocolos CAN y LIN

3.4.2.5. Técnicas de mitigación y *test* on-line

Una vez realizado el análisis de sensibilidad a los SEUs en la red LIN, es conveniente aplicar el endurecimiento a los nodos de la citada red. Los flip-flops que presentaron la mayor sensibilidad frente a los errores SEUs han sido endurecidos con un TMR.

Como se mencionó antes, los registros de configuración fueron la parte más sensible del diseño. Estos registros de configuración representan un 27,32% (56 registros de los 205) de los elementos de memoria del nodo; endureciéndolos con un esquema TMR se consigue una reducción de un 50% en la tasa de averías (del 28,08% al 14,15%). A priori, el aumento en área podría parecer costoso, pero en realidad es la solución más barata aplicada a este nivel de endurecimiento. Reducir la tasa de averías en un nivel superior (por ejemplo, a nivel de sistema) implicaría utilizar otro tipo de redundancias que siendo estas considerablemente más caras.

Con el fin de comparar la eficacia de estas técnicas de mitigación aplicadas a los diseños de los buses LIN o CAN, la Figura 31 muestra el tamaño en términos de FF de los circuitos LIN y CAN endurecidos frente a la tasa de averías. La tasa de avería se representa en el eje horizontal. El eje vertical representa el número de flip-flops del circuito endurecido (como una medida del área de circuito). Esta grafica muestra que si se requiere una alta protección (baja tasa de fallos), el circuito LIN endurecido es más pequeño, mientras que el área del circuito CAN es menor para las tasas de protección bajas. A una tasa de averías del 5%, ambos circuitos tienen el mismo tamaño aproximadamente. Para una tasa de averías del 12%, CAN no requiere ser endurecido, mientras que LIN requiere de 47 flip-flops endurecidos.

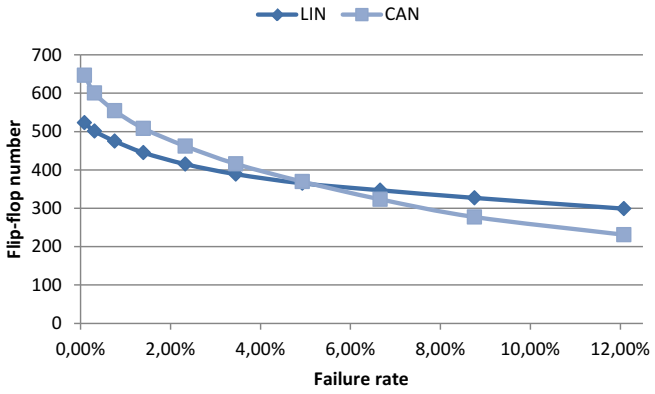


Figura 31. Tasa de averías frente a los biestables endurecidos para los buses CAN y LIN [96]

La principal razón de este comportamiento reside en que los registros de configuración del protocolo LIN son muy sensibles a los fallos, y al endurecerlos adquirimos un aumento de protección alto. Por otra parte, el protocolo CAN es más robusto pero hay muchos flip-flops que tienen que ser triplicados para bajar la sensibilidad global frente a fallos.

Los resultados experimentales muestran que la sensibilidad ante los SEUs de los nodos LIN puede ser reducida con un coste pequeño en área frente a los protocolos más robustos. Resulta recomendable para un primer análisis del endurecimiento del sistema la aplicación de la técnica selectiva TMR.

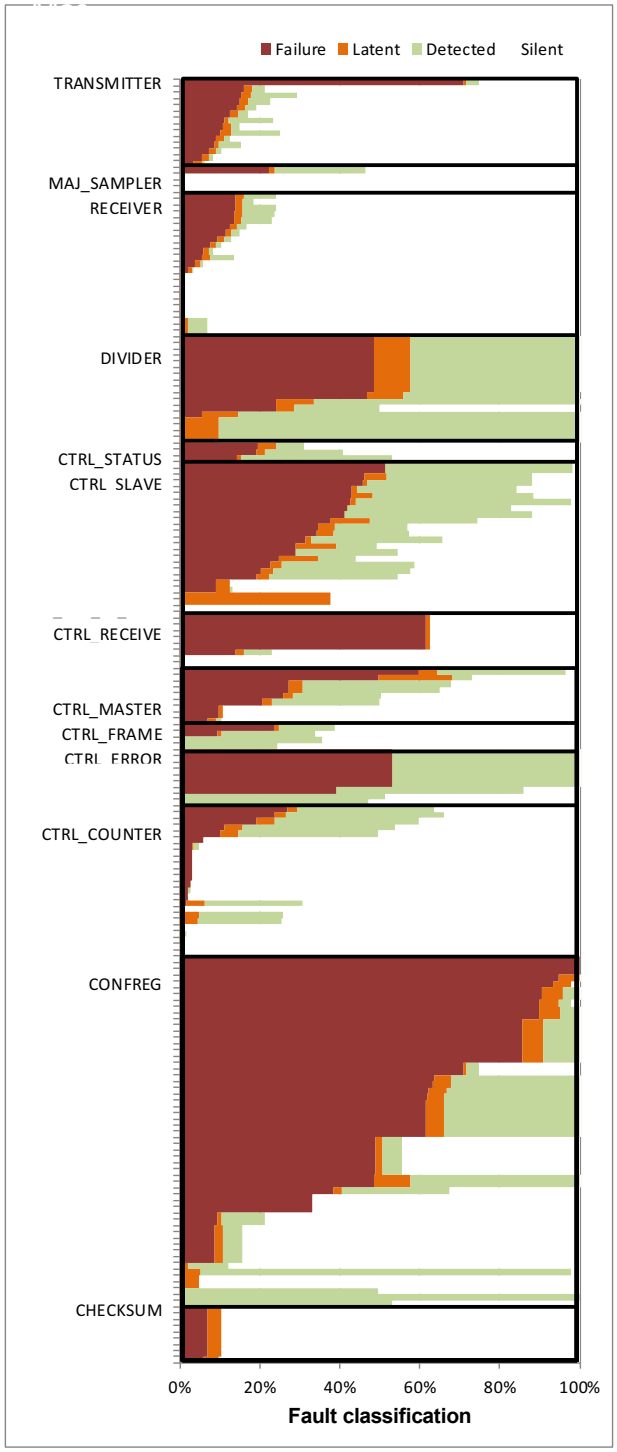


Figura 32. Clasificación de fallos para el bus LIN por bloques [96].

Capítulo 4 PRESENCIA DE LOS FALLOS PERMANENTES EN SISTEMAS DISTRIBUIDOS

Hoy en día, las tecnologías presentes en los circuitos y sistemas electrónicos han sido de gran importancia en la consecución de los niveles de seguridad y robustez requeridos por las distintas aplicaciones. Sin embargo, las citadas tecnologías presentan, en ocasiones, fallos permanentes debidos al envejecimiento de los componentes o a las condiciones adversas del entorno de funcionamiento, como pueden ser la temperatura, la radiación, la velocidad, interferencias electromagnéticas, etc. Debido a ello, resulta importante conocer todos los aspectos del funcionamiento de un sistema electrónico para poder aumentar la controlabilidad y la observabilidad del sistema y así mejorar la detección de fallos permanentes, debido al envejecimiento del mismo.

En este capítulo se utiliza el concepto de fallo permanente para referirnos a aquellos fallos, de efecto permanente, que aparecen durante la vida operativa del circuito debidos al envejecimiento del circuito. No se consideran los fallos debidos a los procesos de fabricación, cuya detección ya está resuelta por las herramientas disponibles en el mercado. La detección de este tipo de fallos debe hacerse de forma concurrente con la operación del circuito analizado (*test on-line*). El objetivo ideal sería detectar el 100% de los posibles fallos debidos a envejecimiento, pero la ausencia de una controlabilidad o una observabilidad completas hace esto casi imposible.

La presencia de los fallos permanentes en los circuitos electrónicos de automoción está regulada por la norma ISO 26262. En particular, de acuerdo con la norma ISO 26262-5 D.2.3.1, los fallos en la unidad del procesamiento y en otros sub-elementos que tienen almacenamiento físico (por ejemplo un registro) o unidades funcionales (por ejemplo decodificadores) tienen que detectarse lo antes posible por medio de código software. Por otra parte, la norma ISO 26262-5 D.1 establece que los fallos tipo "*stuck-at's*" a nivel de puertas lógicas deben cubrirse por el STL.

Como consecuencia de esta normativa, las empresas están buscando soluciones efectivas para realizar las pruebas de sus sistemas en campo *(test on-line*), que sean capaces de cumplir los diferentes requisitos procedentes de las especificaciones del producto, en términos del coste, esfuerzo, tiempo disponible y el consumo. Algunas de estas soluciones se basan en la testabilidad del diseño (DFT: *Design for Testability*) detallada en el Capítulo 2, que normalmente permite al diseñador lograr una buena cobertura de fallos con requisitos de tiempo bastante limitados. Por otro lado, la adopción de soluciones DFT requiere la introducción de algunos cambios en el hardware, lo que no siempre es posible en la práctica, por ejemplo, debido a que los dispositivos proceden por terceros, o porque el coste asociado no es aceptable.

Para algunas aplicaciones específicas, como por ejemplo los protocolos de comunicación, se requiere un enfoque funcional, en el que se estimula el sistema adecuadamente a través de sus entradas funcionales, y se observa su comportamiento, detectando así los posibles fallos. Este enfoque no requiere ningún cambio en el hardware, pero la cobertura que se puede alcanzar con estos fallos depende de la calidad

de los estímulos[2] realizados. El coste del desarrollo de los estímulos puede ser relevante, sobre todo porque todavía no existen herramientas adecuadas para automatizar esta fase en la Automatización del Diseño Electrónico (EDA) (*Electronic Design Automation*).

La robustez de los protocolos de comunicación utilizados en las aplicaciones de automoción es un tema de interés para la comunidad científica y para la industria [89]. En particular, hay varios trabajos que tienen como objetivo mejorar la detección de los fallos en campo y realizar el diagnóstico de estos fallos en las redes de comunicación para automoción como pueden ser CAN, FlexRay o LIN [90]-[96].

En [90], los autores proponen la reutilización de las infraestructuras de depuración disponibles, para realizar pruebas de campo de los sistemas electrónicos incluidos en las aplicaciones de automoción. El enfoque consiste en añadir un componente programable en la Unidad de Control Electrónico (ECU) para probar la gestión de las infraestructuras de depuración. Los experimentos con fallos permanentes en este trabajo requieren tiempos de prueba del orden de minutos.

En los trabajos [91] y [92] se presentan enfoques que se centran en el protocolo CAN. Con el fin de diagnosticar los fallos en los nodos del bus CAN, en [91] se añade un hardware dedicado a monitorizar la transmisión y la recepción con unos contadores de errores. En [92], los autores presentan un método para comprobar el proceso de iniciación en un vehículo: un algoritmo de control de supervisión de la autorización de inicio. Este método se basa en el intercambio de mensajes entre los nodos de la red CAN, comprobando si las respuestas son correctas en un intervalo de tiempo apropiado. No está dirigido a la detección de fallos permanentes o transitorios en los módulos electrónicos, pero sí a la detección de los posibles errores del sistema en el arranque, como, por ejemplo, cuando el botón de inicio no ha sido presionado durante el tiempo suficiente.

El protocolo FlexRay también se ha estudiado ampliamente, ya que se considera como el principal protocolo de comunicación para los sistemas de automoción y de aviónica. En [93], se analiza el comportamiento ante los fallos transitorios de una aplicación que utiliza el protocolo FlexRay y, a partir de los efectos de los fallos observados, se propone un mecanismo para detectar este tipo de errores (diagnóstico) incluyendo una monitorización de errores. En el artículo [94], los autores también se centran en el estudio de las redes construidas con el protocolo FlexRay. La principal idea de estos autores consiste en conectar al bus un bloque comprobador para observar el tráfico de los mensajes e inyectar fallos para estudiar sus efectos. En este caso, el objetivo principal es probar el módulo en lugar de realizar la detección de fallos, pero la ventaja consiste en que este método se puede aplicar en modo *on-line*.

El protocolo LIN tiene menos requisitos de robustez comparado con el FlexRay por lo que hay pocos artículos de investigación de este protocolo. En [95] y [96], se proponen métodos aplicados en la etapa de diseño, pero no hay trabajos con redes LIN que estudien la detección de fallos permanentes en campo. Los autores de [95] presentan un *test* para el protocolo LIN aplicable durante la etapa de diseño. Sin embargo, el creciente número de ECUs en las aplicaciones de automoción actuales, implica la necesidad de construir redes más complejas donde se utiliza el protocolo LIN en subredes no críticas, en lugar del CAN; lo que hace que la robustez del protocolo LIN se esté convirtiendo en una preocupación a la hora de hacer *test on-line* del sistema completo. Aunque los nodos conectados mediante la red LIN en un vehículo no estén realizando tareas críticas para la seguridad de los ocupantes, un fallo en uno de estos

[2] En el documento se utilizarán indistintamente las denominaciones "estímulos", "banco de pruebas", "*workload*", "carga de trabajo" para designar el conjunto de entradas aplicadas a un sistema bajo test.

nodos puede acarrear serias consecuencias; dado que la red LIN está a cargo por ejemplo, del sistema de limpiaparabrisas automáticos, posicionamiento remoto de espejos retrovisores, etc.

En resumen, la mayoría de los estudios existentes de los fallos permanentes en los protocolos de comunicación utilizados en los automóviles, se basan en el uso de unos módulos hardware adicionales o de las infraestructuras de depuración disponibles. Además, el protocolo LIN no ha sido estudiado para este fin, al menos a la hora de escribir esta tesis.

Sabiendo esto, y habiendo estudiado los trabajos anteriores para verificar la presencia de los fallos permanentes en sistemas distribuidos, nos centramos en una red distribuida de nodos conectados por el protocolo LIN. Se utiliza para este estudio una red LIN pequeña pero representativa, para demostrar que mediante la aplicación de varios *tests* de verificación, la mayoría de los fallos permanentes, de tipo *stuck-at,* pueden ser detectados. Por otra parte, en esta tesis doctoral, se realiza un análisis para los fallos permanentes [97] que no pueden ser detectados en campo, lo que lleva a la identificación de fallos no comprobables [98] con el funcionamiento *on-line*. Así, se proponen pautas para mejorar los estímulos de prueba con el fin de aumentar la cobertura de fallos permanentes en tiempo de funcionamiento (*test on-line*).

4.1. CAMPAÑA DE INYECCIÓN DE FALLOS PERMANENTES

Como se ha dicho anteriormente, la inyección de fallos en los circuitos bajo prueba mediante emulación de hardware, en dispositivos reconfigurables (FPGAs), se ha convertido en una solución común para el estudio de la sensibilidad durante las primeras etapas del ciclo de diseño. Hay muchas herramientas que realizan campañas de inyección de fallos transitorios en los circuitos descritos en VHDL o *soft cores* a través de la emulación en las FPGAs [99][100][101]. Estas campañas tienen como objetivo testear el efecto de la radiación ionizante y comprobar las capacidades de *test on-line* del circuito. La identificación temprana de los elementos débiles en el circuito, así como la evaluación de las técnicas de endurecimiento insertadas son las principales ventajas proporcionadas por estas herramientas.

La técnica de detección de fallos permanentes en campo ha sido estudiada recientemente como un método eficaz para detectar el envejecimiento de los dispositivos electrónicos. La ejecución de una carga de trabajo (*workload*) específica en el arranque de los sistemas y subsistemas electrónicos en muchas aplicaciones (equipos médicos, aplicaciones industriales, automóviles, etc.) puede ayudar a detectar fallos permanentes causados por dicho envejecimiento.

Pero, debido a la complejidad del sistema las cargas de trabajo eficientes son difíciles de generar y analizar, ya que no todos los pines de los elementos del sistema son accesibles. Los *tests* de fabricación se desarrollan cuidadosamente teniendo en cuenta que cada salida del circuito debe ser continuamente observable. Dichos *tests* presentan muchas restricciones en cuanto al control y la observación de las entradas y salidas del circuito. Principalmente se imponen dos requisitos en la mayoría de las aplicaciones. En primer lugar, la carga de trabajo debe ser la única herramienta para detectar los fallos permanentes (no hay estructuras de prueba internas que puedan ser utilizadas). En segundo lugar, aunque se proponen soluciones para los sistemas basados en microprocesadores, algunos enfoques se centran en protocolos de comunicación de redes de sistemas distribuidos. Por lo tanto, la industria requiere algoritmos y herramientas que ayuden a la hora de generación de estas cargas de trabajo específicas, y en el mundo académico se está estudiando la aplicación de la mejor solución al citado problema.

En este sentido, la inyección de fallos permanentes a través de la emulación de hardware, para la evaluación de las capacidades del circuito es actualmente una realidad en muchas aplicaciones industriales.

4.4.1. Herramienta de inyección de fallos permanentes

La emulación autónoma es una herramienta para la evaluación de la sensibilidad de los circuitos digitales a través de la inyección de fallos en los elementos del circuito. Esta herramienta inyecta fallos transitorios en los elementos de memoria y esta descrita con mayor detalle en el apartado 2.3. En esta tesis doctoral, se han ampliado las prestaciones de esta herramienta, incluyendo la capacidad de inyección de los fallos permanentes en cada *flip-flop* y en cualquier ciclo de reloj de la carga de trabajo[3].

En esta herramienta, se prototipan dos copias de los elementos de memoria del circuito bajo *test* en una FPGA, *golden* y *faulty*. La ejecución en paralelo de las dos versiones del circuito permite la comparación de las salidas del circuito ciclo a ciclo y en todos los elementos de la memoria interna. Los fallos inyectados se clasifican por el efecto producido en el comportamiento del sistema de acuerdo a las siguientes categorías:

1. En el caso de que el efecto de los fallos se propague a las salidas:
   a. Un fallo se clasifica como *Detectado* cuando algunos de los mecanismos de detección de errores implementado en el módulo es capaz de detectar el fallo (por ejemplo, un *flag* de interrupción se activa cuando no se espera un valor concreto o viceversa).
   b. Un fallo se clasifica como *Avería* cuando se realiza una transmisión defectuosa pero el modulo no detecta ningún mal comportamiento (por ejemplo los datos se corrompen pero la paridad y el *checksum* son correctos).
2. En el caso de que el efecto de los fallos no se propague a las salidas:
   a. Un fallo se clasifica como *Silencioso* si su efecto ha desaparecido por completo del circuito.
   b. Un fallo se clasifica como *Latente* si sus efectos se están almacenando en algunos elementos de memoria del circuito.

La inyección de los fallos permanentes en la lógica combinacional es necesaria y posible, por medio de las herramientas desarrolladas dentro del grupo DMA de la UC3M; pero las herramientas de simulación para los *test* de fabricación, proporcionadas por los fabricantes de herramientas CAD (Synopsys® entre otros) aceleran enormemente el proceso, dada la gran cantidad de puertas lógicas de los circuitos electrónicos que se están considerando en este trabajo de tesis doctoral.

El uso de la herramienta de emulación hardware para analizar el efecto de los fallos permanentes en los circuitos bajo prueba, permite un análisis rápido (millones de fallos por segundo) del efecto de los fallos permanentes en los elementos de control (máquinas de estado, registros de configuración, etc.) que se ve complementado con la simulación de fallos en los bloques combinacionales, con herramientas de simulación de *test* de fabricación.

4.4.2. Sistema bajo test de inyección de fallos

El sistema utilizado para evaluar la metodología de inyección de fallos permanentes es un sistema distribuido con varios nodos interconectados mediante el

[3] Aunque la inyección más común sea en el primer ciclo de reloj del banco de pruebas

protocolo LIN. Para testear el protocolo se conectan dos nodos LIN (un maestro y un esclavo) aunque se puede conectar hasta 15 nodos esclavos a la red[4].

La carga de trabajo funcional ejecutada incluye once pruebas diferentes. En primer lugar, hay cuatro tests básicos de transmisión de tramas de maestro a esclavo. Las tres primeras pruebas envían 2, 4 y 8 bytes, respectivamente, mientras que la cuarta prueba envía un dato de identificación y el esclavo debe responder. Estas pruebas básicas están destinadas a verificar que los dos nodos se comunican correctamente. Posteriormente, hay otros *tests*, que comprueban la presencia de error de paridad, la detección de errores del *checksum* y el error de la trama; estos *tests* requieren forzar el bus para inyectar los fallos. El esquema de prueba típico consiste en escribir en un registro de configuración, esperar una interrupción, y entonces actuar, leyendo el byte recibido o comprobando los distintos registros de estado.

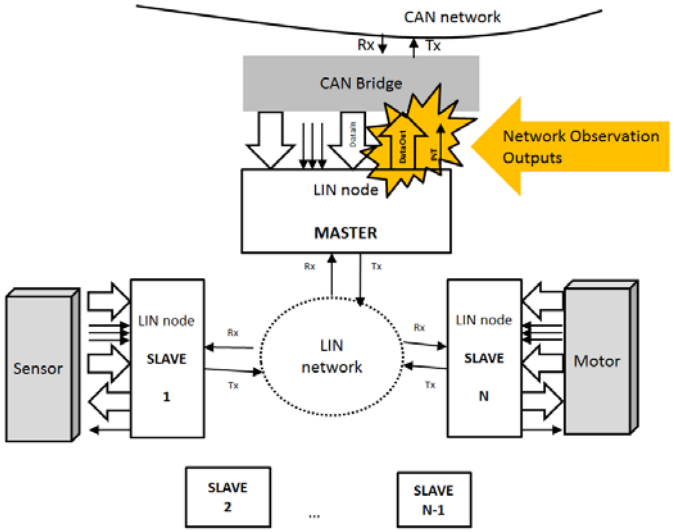


Figura 33. Sistema distribuido (LIN) utilizado para la inyección de fallos permanentes [97]

La inyección de los fallos permanentes es un problema sencillo, utilizando la herramienta de emulación autónoma desarrollada en el grupo DMA de UC3M. Sin embargo, la detección de errores debidos a la presencia de estos fallos permanentes es compleja si consideramos el sistema distribuido anteriormente descrito. En una red LIN, los esclavos son difícilmente observables desde el punto de vista de niveles jerárquicos superiores (desde la unidad de control, situada en una red CAN y que se conecta al maestro de la red LIN). Los nodos esclavos están conectados a una carga útil, mediante una interfaz de datos y direcciones y una línea de interrupción. El nodo maestro no tiene comunicación con los nodos esclavos, excepto con las señales del bus LIN (Rx / Tx). Si tenemos en cuenta que la detección de errores, debidos a fallos permanentes, se tiene que hacer a través del nodo maestro (el único que permite la lectura y escritura desde los niveles superiores), la controlabilidad y la observabilidad de los elementos internos de los nodos esclavos de la red LIN están muy limitadas y, por tanto, su testabilidad es difícil. Será necesario proponer soluciones para aumentar estas dos propiedades en los nodos esclavos, para asegurar una detección aceptable de fallos permanentes.

4.4.3. Campaña de inyección de fallos

En el sistema distribuido, basado en protocolo LIN, anteriormente descrito, se ha inyectado un fallo permanente en cada *flip-flop* (410 FF) y al inicio de los 37.381 ciclos de reloj de carga de trabajo. Sólo los elementos secuenciales en el circuito se han

[4] El nodo básico LIN está tomado de la referencia [76] de Xilinx, que incluye un banco de pruebas básico para las principales funciones del nodo maestro-esclavo LIN

considerado para este estudio. Se han establecido como puntos de observación el bus de datos y las salidas que se observan con un indicador de la interrupción (INT) desde el nodo principal.

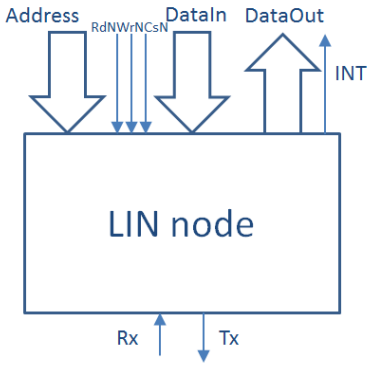


Figura 34. Las salidas observables en el nodo típico del LIN network [97]

Las siguientes Tabla 21 y Tabla 22 muestran los resultados para el nodo maestro, de una campaña de inyección de fallos tipo *stuck-at-0* y *stuck-at-1*. Las tablas muestran la clasificación de los resultados en función de los fallos y distintos bloques del módulo controlador del bus en los que se han inyectado los fallos. Los bloques que componen un nodo básico en el protocolo LIN son los registros de configuración, el bloque de control, el divisor de frecuencia, los bloques de recepción (receptor y *majority sampler*), el transmisor y el generador/comprobador de *checksum,* Figura *29* en el capítulo 3.

	#FF	**Avería**	**Latente**	**Silencioso**
CONFREG	56	27	0	29
CRL_CHECKSUM	2	0	0	2
CRL_COUNTER	22	19	0	3
CRL_ERROR	9	3	0	6
CRL_FRAME	4	4	0	0
CRL_MASTER	9	9	0	0
CRL_SLAVE	23	21	2	0
CRL_RECEIVE	9	7	1	1
CRL_SER_IN_LAST	1	0	1	0
CRL_STATUS	3	3	0	0
DIVIDER	17	6	1	10
RECEIVER	23	23	0	0
SAMPLER	4	4	0	0
TRANSMITTER	14	13	0	1
CHECKSUM	9	0	0	9
Total	*205*	139 67,80%	5 2,44%	61 29,76%

Tabla 21. La clasificación de los fallos para nodo Maestro stuck-at-0

	#FF	Avería	Latente	Silencioso
CONFREG	56	52	0	4
CRL_CHECKSUM	2	0	2	0
CRL_COUNTER	22	18	3	1
CRL_ERROR	9	9	0	0
CRL_FRAME	4	4	0	0
CRL_MASTER	9	9	0	0
CRL_SLAVE	23	21	0	2
CRL_RECEIVE	9	7	1	1
CRL_SER_IN_LAST	1	1	0	0
CRL_STATUS	3	3	0	0
DIVIDER	17	17	0	0
RECEIVER	23	23	0	0
SAMPLER	4	4	0	0
TRANSMITTER	14	14	0	0
CHECKSUM	9	9	0	0
Total	*205*	191 93,17%	6 2,93%	8 3,90%

Tabla 22. La clasificación de los fallos agrupados para nodo Maestro stuck-at-1

Como resultado global, con la carga de trabajo funcional anteriormente descrita, alrededor del 93% de fallos *stuck-at-1* se detectan en los elementos de memoria del nodo maestro LIN, y alrededor de 68% de fallos *stuck-at-0*. Esta diferencia en los resultados se debe principalmente al valor de inicialización de la mayoría de los flip-flops (inicialización con el *reset* a '0'), que ayuda en la detección de *stuck-at-1*. Con respecto a las otras clasificaciones de los fallos, alrededor de 30% de los fallos *stuck-at-0* no provocan valores diferentes del estado (se consideran como *Silencioso*), mientras que alrededor del 2% de ellos permanecen latentes en el circuito. Este resultado es diferente respecto a los fallos *stuck-at-1*, donde la controlabilidad es mejor (sólo el 4% de los fallos son silenciosos), pero la observabilidad es ligeramente peor (3%).

Analizando en detalle la clasificación de los fallos en cada bloque del nodo maestro LIN, se destacan tres bloques donde ningún fallo *stuck-at-0* se detecta en las salidas de nodos, todos ellos están relacionados con el cálculo de *checksum*. Incluso, en uno de ellos (en el bloque de control que está relacionado con el *checksum*) no se detecta ningún fallo *stuck-at-1*. Pero se destacan los bloques con peores resultados que son los registros de configuración (sólo 48% de los fallos permanentes *stuck-at-0* se detectan) y el divisor de frecuencia (sólo 35% de los fallos permanentes *stuck-at-0* se detectan) entre los que el bloque de control no figura. Con respecto a los fallos *stuck-at-1* una mejor capacidad de detección se observa con esta carga de trabajo, aunque cuatro *flip-flops* en el bloque de registros de configuración permanecen sin ejercitar, que se clasifican como fallos *Silenciosos*.

Como se ha mencionado antes, la norma ISO 26262 para aplicaciones de automoción requiere que el 90% de los fallos permanentes se detecten incluso cuando el Nivel Integral de Seguridad en el Automoción es B, que denota un bajo nivel de seguridad requerido. Por lo tanto, la mejora de esta carga de trabajo funcional es obligatoria, especialmente para los fallos *stuck-at-0*.

Por otro lado, si analizamos la clasificación de los fallos de los nodos esclavos, se obtienen peores resultados. Los registros de configuración no son accesibles directamente a través del bus LIN. Por lo tanto, los efectos de los fallos inyectados en estos elementos sólo son observables a través de un mal funcionamiento en el procesamiento de comandos (decodificación de la trama en bloque de control). La Tabla 23 y la Tabla 24 detallan los resultados para el nodo esclavo. Sólo alrededor del 16% de

los fallos *stuck-at-0* inyectados se detectan y alrededor del 22% de los fallos inyectados *stuck-at-1*.

	#FF	Avería	Latente	Silencioso
CONFREG	56	8	9	39
CRL_CHECKSUM	2	1	0	1
CRL_COUNTER	22	5	1	16
CRL_ERROR	9	1	0	8
CRL_FRAME	4	1	1	2
CRL_MASTER	9	2	0	7
CRL_SLAVE	23	3	1	19
CRL_RECEIVE	9	0	2	7
CRL_SER_IN_LAST	1	0	0	1
CRL_STATUS	3	0	1	2
DIVIDER	17	1	3	13
RECEIVER	23	3	0	20
SAMPLER	4	1	0	3
TRANSMITTER	14	4	0	10
CHECKSUM	9	2	0	7
Total	205	32 15,61%	18 8,78%	155 75,61%

Tabla 23. Clasificación de los fallos agrupados para el nodo esclavo, stuck-at-0

	#FF	Avería	Latente	Silencioso
CONFREG	56	13	0	43
CRL_CHECKSUM	2	1	0	1
CRL_COUNTER	22	4	0	18
CRL_ERROR	9	4	0	5
CRL_FRAME	4	0	0	4
CRL_MASTER	9	3	0	6
CRL_SLAVE	23	4	0	19
CRL_RECEIVE	9	2	0	7
CRL_SER_IN_LAST	1	1	0	0
CRL_STATUS	3	0	0	3
DIVIDER	17	3	0	14
RECEIVER	23	5	0	18
SAMPLER	4	1	0	3
TRANSMITTER	14	2	0	12
CHECKSUM	9	2	0	7
Total	205	45 21,95%	0 0,00%	160 78,05%

Tabla 24. Clasificación de los fallos agrupados para el nodo esclavo, stuck-at-1

4.4.4. Comprobación eficiente de fallos permanentes en el arranque del sistema

En los sistemas electrónicos para las aplicaciones distribuidas de control o monitorización de múltiples subsistemas, por ejemplo de la automoción, es necesario utilizar unas cargas de trabajo eficientes, que permitan el funcionamiento correcto de forma permanente; esto es, que no se debe sobrecargar el sistema con pruebas continuas de comprobación *on-line* de errores o fallos. Sin embargo, la mejor forma de detectar la presencia de errores y/o fallos es utilizar el enlace de comunicaciones entre todos los nodos y, a través suyo, realizar tareas de comprobación del máximo número de elementos de los nodos de la red. La realización de pruebas en el arranque del sistema es un método aceptado y utilizado por numerosos fabricantes y desarrolladores de este tipo de sistemas. La generación de un *workload* de este tipo, en las primeras etapas del ciclo de diseño, y que tenga en cuenta la controlabilidad y la observabilidad de cada

nodo en la red es muy importante para asegurar una alta tasa de detección de fallos y, por tanto, un nivel de seguridad adecuado.

4.4.4.1. Generación de una carga de trabajo eficiente

Los principales elementos necesarios para la generación de una carga de trabajo eficiente, en la detección *on-line* de fallos permanentes, son:

• *Netlist* o descripción del circuito, listo para ser simulado o emulado.

• Herramienta de inyección de fallos, capaz de inyectar fallos permanentes en los elementos internos de los nodos de la red.

• Una carga de trabajo típica que permita comprobar el funcionamiento de la red y sus nodos. Esta carga de trabajo debe comprobar la funcionalidad principal de la red teniendo en cuenta todos los modos de funcionamiento del sistema, modos de error, configuración y comandos de intercambio de datos.

A continuación, se describe la metodología, que se ha propuesto en esta tesis doctoral, para lograr una alta cobertura de fallos, y que se compone de cuatro pasos:

1. Campaña de inyección de fallos permanentes sobre una carga de trabajo funcional, descrita en lenguaje VHDL o Verilog, proporcionada por el fabricante o desarrollada por un ingeniero, de acuerdo con las especificaciones del circuito o de la red.

2. Clasificación de los fallos de *stuck-at-0* y *stuck-at-1*. Clasificación de los fallos calculados e identificación de fallos no detectables.

3. Localización de los elementos que no cubre el *workload* aplicado. Identificación de los fallos latentes y silenciosos.

   a. *Fallos latentes,* se deben a que se inyecta un fallo pero no hay propagación de su efecto observable en las salidas. Los elementos internos del circuito difieren de los del circuito sin fallo inyectado. Para este tipo de fallos, la observabilidad debe estar asegurada.
   b. *Fallos silenciosos,* se deben a que el fallo no ha provocado ningún efecto en el circuito, ni en las salidas ni en los elementos internos del circuito se detectan diferencias con respecto al circuito sin fallo inyectado.

4. Re-generación de una nueva carga de trabajo que asegura la inyección de todos los fallos posibles y la observación de sus efectos donde se realiza la clasificación de su efecto.

Una vez aplicados estos pasos de nuevo se repiten los pasos de 1 a 4 hasta que la cobertura de fallos alcanza el 90% de *stuck-at-0* y *stuck-at-1*.

Completando el *workload* se asegura la inyección de fallos en los elementos internos que producen fallos silenciosos, para este fin se identifican y se activan los procedimientos de escritura para estos elementos de memoria, según las especificaciones del protocolo.

Para aquellos elementos que producen fallos latentes se mejora la observabilidad generando alguna lectura de los procedimientos de registros internos, comprobando los divisores de frecuencia, máquinas de estados y mecanismos de detección de error.

Después de varias iteraciones, se puede comprobar que hay algunos fallos que no son testeables para el sistema. Para estos fallos hay dos opciones. En primer lugar, no utilizar estos bloques afectados durante la carga de trabajo típica, porque estos fallos permanentes no tienen importancia. En segundo lugar, los bloques afectados no son visibles dentro del sistema analizado (red distribuida donde sólo algunos elementos de los nodos son visibles mediante el enlace de comunicaciones). En este último caso, el comportamiento correcto del sistema también se asegura, desde el punto de vista del funcionamiento de la red.

4.4.4.2. Carga de trabajo eficiente para detectar fallos permanentes en un sistema distribuido, basado en comunicación con una red LIN

Se ha aplicado la metodología anteriormente detallada a la red LIN básica descrita en el apartado 4.4.2. Los resultados experimentales de la inyección de fallos por emulación en el sistema distribuido con comunicación mediante protocolo LIN, muestran que la cobertura de fallos del nodo maestro por *stuck-at-0,* con una carga de trabajo funcional, es más baja de la que requiere la norma (ISO-26262, ASIL nivel B, 90%), mientras que la cobertura de fallos por *stuck-at-1* es mayor que 90% con lo cual responde a los requerimientos de la norma. Para los nodos con la funcionalidad esclava, la cobertura de fallos es mucho menor.

Al aplicar la metodología que se detalla en la sección 4.4.4.1, para identificación de los fallos latentes y silenciosos, e incluyendo procedimientos de escritura y lectura de los comandos en el *workload* original, así como activando en el *workload* varios modos de configuración para divisores de frecuencia y otros registros de configuración, se obtienen resultados mejorados de la cobertura de fallos. Para este sistema de nodos con el protocolo LIN se ha realizado sólo una iteración. Partiendo del *workload* inicial (37.381 ciclos de reloj) se ha añadido 6.776 vectores de prueba adicionales. Los nuevos resultados de la clasificación se muestran en la Tabla 25 y Tabla 26 para el nodo maestro. En este nodo maestro se detecta la mayoría de los fallos *stuck-at-1* (96%), mientras que todavía hay elementos en los registros de configuración que no se controlan para el *stuck-at-0* con este *workload*. La cobertura de fallos por *stuck-at-0* es del 89%. Se puede realizar más iteraciones con el fin de mejorar la cobertura de fallos.

	#FF	**Avería**	**Latente**	**Silencioso**
CONFREG	56	43	1	12
CRL_CHECKSUM	2	2	0	0
CRL_COUNTER	22	20	0	2
CRL_ERROR	9	8	0	1
CRL_FRAME	4	3	1	0
CRL_MASTER	9	9	0	0
CRL_SLAVE	23	23	0	0
CRL_RECEIVE	9	7	0	2
CRL_SER_IN_LAST	1	1	0	0
CRL_STATUS	3	3	0	0
DIVIDER	17	17	0	0
RECEIVER	23	21	0	2
SAMPLER	4	4	0	0
TRANSMITTER	14	14	0	0
CHECKSUM	9	8	0	1
Total	205	183 89,27%	2 0,98%	20 9,76%

Tabla 25. La clasificación de los fallos agrupados para nodo Maestro stuck-at- 0 con banco de pruebas mejorado

Con un *workload* funcional se detectan fallos permanentes en los nodos que tienen funcionalidad de nodo maestro. Estos fallos permanentes se deben al envejecimiento de dispositivo.

Por otro lado, para los nodos con la funcionalidad de esclavo, la mejora del número de fallos detectados requiere, aparte de un *workload* funcional, la aplicación también de otras técnicas. En general, para muchos circuitos complejos con funcionalidad distribuida, debido a la inicialización previa de los registros, los fallos *stuck-at-0* son más difíciles de detectar que los *stuck-at-1*. El uso de una herramienta de inyección de fallos en las primeras etapas del ciclo de diseño ayudará a detectar los posibles efectos del envejecimiento en el sistema final. Para los fallos no comprobables,

que no son accesibles debido a la arquitectura de la red, se deben proponer soluciones adicionales. En este sentido, los nodos esclavos más críticos se pueden comprobar externamente con técnicas de monitorización de firma.

	#FF	**Avería**	**Latente**	**Silencioso**
CONFREG	56	54	2	0
CRL_CHECKSUM	2	2	0	0
CRL_COUNTER	22	22	0	0
CRL_ERROR	9	9	0	0
CRL_FRAME	4	2	1	1
CRL_MASTER	9	8	1	0
CRL_SLAVE	23	23	0	0
CRL_RECEIVE	9	9	0	0
CRL_SER_IN_LAST	1	1	0	0
CRL_STATUS	3	3	0	0
DIVIDER	17	17	0	0
RECEIVER	23	22	1	0
SAMPLER	4	4	0	0
TRANSMITTER	14	13	0	1
CHECKSUM	9	9	0	0
Total	205	198 96,59%	5 2,44%	2 0,98%

Tabla 26. La clasificación de los fallos agrupados para nodo Maestro, stuck-at-1 con banco de pruebas mejorado

Se propone un método [112] para generar *tests* funcionales de detección de los fallos permanentes en las redes distribuidas. Este método requiere los siguientes elementos:

- Descripción del circuito.
- Una carga de trabajo funcional.
- Herramientas de análisis de fallos: herramienta de inserción de fallos permanentes y un simulador digital con indicadores de cobertura del código (como por ejemplo Modelsim).

La Figura 35 describe el método propuesto, que consta de los siguientes pasos:

Paso 1: Se realiza una simulación funcional con el fin de obtener las métricas relacionadas con la cobertura del código: los estados de la FSM, las tramas, las condiciones y expresiones, la cobertura de las conmutaciones, etc. Este análisis marca las partes del diseño que no están siendo ejecutadas desde la herramienta CAD, y señala las partes específicas del código que no han sido cubiertas. Si la cobertura del código no se puede mejorar, debido a las características del diseño o restricciones puestas por la aplicación, el siguiente paso es el Paso 3.

Paso 2: Modificar el banco de pruebas teniendo en cuenta las restricciones de la aplicación.

Paso 3: Se realiza una simulación de los fallos permanentes mediante la que se obtiene la clasificación de los fallos y la cobertura del *test*. Se utiliza el modelo *stuck-at*. Los fallos permanentes que no se pueden comprobar se identifican por el simulador de averías y nunca se detectan mediante los *tests* funcionales.

Paso 4: Además de los fallos estructuralmente no comprobables, identificados por el simulador de averías, es necesario identificar los fallos que son funcionalmente no comprobables (FUT: *Functionally Untestable*).

Paso 5: Los fallos FUT identificados en el paso 4 se eliminan de la lista de los fallos no detectados y un nuevo valor de cobertura del *test* (cobertura del *test* funcional) se

calcula. Esta métrica representa, de una manera más precisa, la cobertura del *test* que se puede alcanzar durante un *test on-line* sin modificaciones del hardware del sistema. Si la cobertura del *test* funcional es mayor que un valor exigido, el banco de pruebas es óptimo para realizar los *tests* funcionales en el campo; de lo contrario, el banco de pruebas debe de ser modificado mediante la inclusión de nuevos estímulos que deben centrarse en la activación y la observación de los fallos no detectados (el paso 2).

El diagrama de flujo de la metodología propuesta se muestra en la Figura 35.

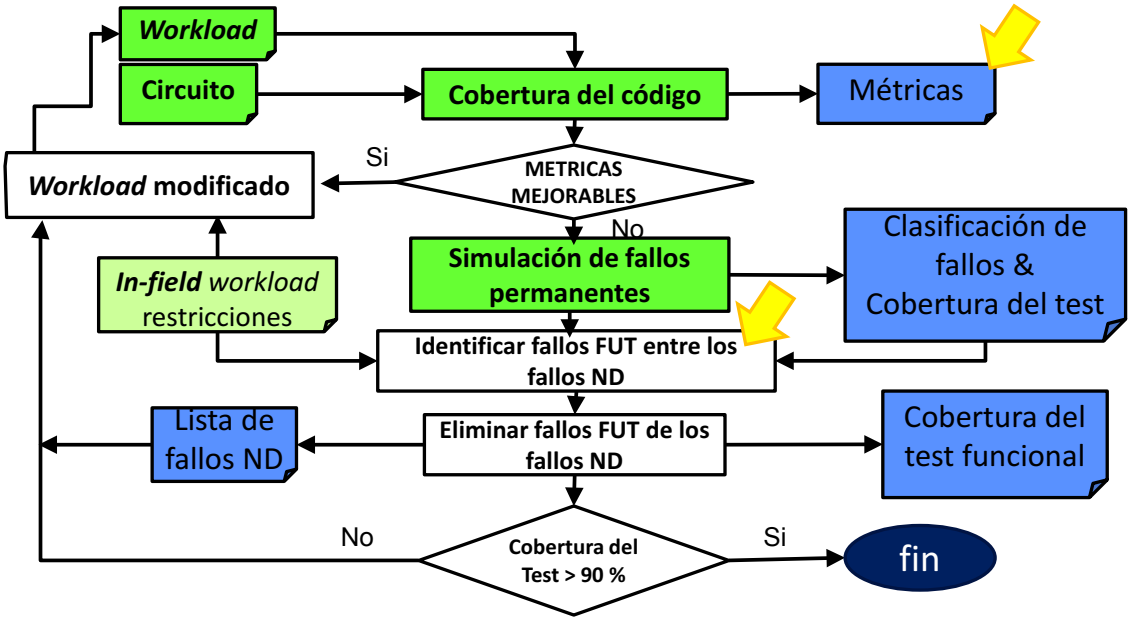


Figura 35. Descripción del método propuesto de generación del *test* funcional para los sistemas distribuidos [112]

La identificación de los fallos FUT es una tarea principal con el fin de evaluar el banco de pruebas. Algunos de estos fallos dependen de la aplicación, pero también hay otros fallos FUT que afectan al hardware de la comunicación. Estos fallos son comunes en los sistemas de hardware distribuidos. Los fallos no detectados están relacionados con el *reset* asíncrono. Esta señal no se puede detectar funcionalmente ya que no es posible ni conveniente activar el *reset* asíncrono de forma *on-line*. En los sistemas de hardware distribuidos, hay que considerar también que los fallos no comprobables funcionalmente pueden estar relacionados con la configuración del nodo, el modo maestro / esclavo en cada nodo o con el *baudrate*, las señales que afectan a la lógica de comprobación del error, ya que generalmente resulta imposible, forzar errores en el *test on-line* para comprobar tal lógica.

Con el fin de identificar los fallos FUT, en la lista de los fallos proporcionada por el simulador de averías permanentes, se puede utilizar un script a medida para encontrar el *reset* y los flip-flops relacionados con la funcionalidad determinada y generar el flujo de datos que permite encontrar la lógica combinacional relacionada.

Capítulo 5 TÉCNICAS DE AUMENTO DE LA TESTABILIDAD ON-LINE PARA SISTEMAS DISTRIBUIDOS DIGITALES. VALIDACIÓN MEDIANTE CAMPAÑAS DE IRRADIACIÓN

De acuerdo a lo explicado en los capítulos anteriores, se ha podido observar que la realización del chequeo continuo, en tiempo de funcionamiento, de los sistemas electrónicos digitales distribuidos se ve invalidada por las bajas tasas de observabilidad y controlabilidad de los nodos de la red distribuida. De hecho, se ha visto en el capítulo 3 que incluso para los protocolos de comunicación con alta robustez, es difícil alcanzar los niveles mínimos de seguridad requeridos por los estándares. Esto es aplicable tanto a los fallos transitorios (radiación ionizante y otras fuentes de fallos) como a los fallos permanentes debidos al envejecimiento del circuito.

Así mismo, incluso en los *test* de validación de la robustez frente a los fallos transitorios, realizados mediante emulación *hardware* o mediante ensayos de irradiación con aceleradores de partículas, es difícil determinar la presencia de fallos porque no siempre los errores provocados son visibles externamente, aunque no se pueda asegurar que estos últimos no existan [102].

En este capítulo se proponen técnicas que permiten mejorar la observabilidad y la controlabilidad de un sistema electrónico digital distribuido, con el propósito de aumentar la calidad del diagnóstico realizado, tanto durante la etapa de diseño y calificación de los sistemas, como en la etapa de funcionamiento en el campo.

La validación de estas técnicas ha tenido su pleno desarrollo con el diseño de un sistema distribuido completo, el análisis de su robustez mediante técnicas de inyección de fallos mediante emulación *hardware*, la inserción de estructuras tolerantes a fallos y elementos de detección *on-line* de la presencia de fallos y la realización de las campañas de inyección de fallos mediante irradiación con un haz de protones en un acelerador de partículas, ciclotrón [103]. De esta forma, se ha podido proponer y validar una metodología de verificación completa de un sistema distribuido sometido a la irradiación; que es, por supuesto, extrapolable a cualquier otro método de inyección de fallos forzado que se utilice para determinar la robustez de dicho sistema distribuido, y aplicable a la detección *on-line* de fallos transitorios y permanentes en el funcionamiento en campo de dicho sistema.

En las campañas de irradiación, se puede observar de forma continua el funcionamiento del sistema bajo prueba, permitiendo la realización de un informe *on-line* de los fallos debidos a los efectos de SEUs/SEFIs que pueden aparecer en cualquier elemento de la red, el chequeo de la capacidad de recuperación del sistema electrónico distribuido y sus componentes y la calificación de cómo de robusto es el sistema distribuido.

5.1. VERIFICACIÓN Y ENDURECIMIENTO DE SISTEMAS ELECTRÓNICOS DISTRIBUIDOS

Como se ha mencionado anteriormente, la medida de la fiabilidad de los sistemas electrónicos distribuidos debe incluir estudios de todos los efectos que pueden

producirse en un sistema incluyendo los efectos de radiación, envejecimiento, ruido, interferencias electromagnéticas (EMI), retrasos en el tiempo de comunicación, etc. Si el funcionamiento normal del sistema distribuido, sometido a un entorno crítico, sufre de la aparición de fallos tipo SEU o SET, se hace necesaria la aplicación de técnicas de endurecimiento colaborativo [111]. El endurecimiento colaborativo consiste en la utilización de elementos no robustos en un sistema robusto, gracias a la cuidadosa distribución redundante de las tareas críticas entre varios elementos, que son capaces de mantener la tasa de fallos en niveles muy bajos porque están corrigiéndose continuamente, utilizando técnicas de votación mayoritaria.

Para endurecer los sistemas electrónicos digitales distribuidos, de modo similar a los sistemas no distribuidos, lo más común es utilizar algún tipo de redundancia. Así, si se detectan las tareas críticas y sólo se aplican las técnicas de redundancia a estas tareas más vulnerables, se puede disminuir en gran medida los altos costes de implementación de la redundancia.

Dado que los fallos de *hardware* afectan al sistema en diferentes niveles, las técnicas de tolerancia a fallos pueden implementarse en cada nivel. A continuación, se presentan las estrategias posibles para cada nivel, y se describen las ventajas e inconvenientes de cada una de ellas.

1. Redundancia de hardware físico.

Cualquier sistema electrónico digital distribuido tiene múltiples sensores y actuadores, que pueden fallar durante el tiempo del funcionamiento. Para aumentar la confiabilidad del sistema se usa la replicación de este *hardware* físico. Muchas veces un sistema complejo, de este tipo, se diseña con múltiples sensores y actuadores que pueden tener capacidades complementarías duplicando algunos sensores y actuadores, o teniendo distintos sensores que realicen la misma medida. Normalmente, es un tipo de sistemas que trabajan en condiciones extremas y deben cumplir restricciones de tamaño y peso.

2. Redundancia hardware del circuito electrónico.

En el sistema distribuido, se replican elementos electrónicos, y en concreto los bloques identificados como más críticos a nivel del protocolo, en el control o en otras funcionalidades. Para ello, se aplican los métodos de redundancia de hardware descritas en el capítulo 2. La detección de hardware con fallo se puede realizar incluso con una aplicación software, mediante el código de votación de la mayoría.

3. Componentes robustos

En el sistema distribuido, se pueden utilizar componentes calificados para aplicaciones muy críticas (*rad-hard*) en aquellos elementos de la red que se han identificado como los más sensibles o cuya funcionalidad es crucial en el funcionamiento del conjunto. La utilización de este tipo de componentes es costosa y no siempre posible (no hay componentes *rad-hard* de todos los dispositivos, el coste es elevado, no está permitido su uso en algunas aplicaciones, etc.).

4. Comportamiento robusto de estrategias redundantes.

Este método requiere la implementación de la redundancia en los niveles más altos del control. Para este fin se diseñan varias técnicas para realizar la misma tarea. Si

el rendimiento es peor de lo esperado se aplica otra estrategia de control para la misma tarea y se comparan los rendimientos de distintas técnicas. Este método se puede aplicar hasta que se encuentre la técnica con el rendimiento aceptable para el controlador, en lugar de modificar la misma una y otra vez sin éxito.

Este método no se dedica a la causa del problema, solo se activa en caso de ocurrir un problema. Esto hecho podría ser perjudicial para un sistema que debe funcionar en un entorno crítico. El correcto funcionamiento del sistema tiene que detectar, enmascarar y recuperar los errores en el bajo nivel del sistema de control.

5. Sistema robusto global.

La tolerancia ante los fallos comprende las siguientes cuatro fases: detección de errores, enmascaramiento, recuperación y reintegración. Los fallos locales se detectan dentro del sistema de la red a bajo nivel. Si después de detectar dichos fallos, estos se enmascaran, ya no afectarán al nivel más alto del sistema. Los fallos globales inevitablemente afectan a nivel global del sistema. Pueden detectarse en el control de bajo nivel y se compensan dentro del control de alto nivel, o detectarse y compensarse en el control de alto nivel.

A continuación se enumeran las técnicas de verificación y endurecimiento de circuito a nivel local y global que se han analizado en esta tesis doctoral, para su posible aplicación en un sistema distribuido.

1. ***Duplicación de nodo***. Esta técnica permite avisar de la presencia de un fallo, aunque no se corrige su efecto.
2. ***Triple replicación del nodo***. Aplicación del TMR (Triple Modular Redundancy). Se realiza la replicación parcial o completa del nodo. Esta técnica es capaz de detectar y corregir fallos simples en cualquiera de las réplicas. En todo momento las réplicas de cada elemento tienen que estar en el mismo estado, analizar las mismas entradas y proporcionar las mismas salidas. Se debe:
    a. Aplicar el bloque de votación de mayoría a los bloques replicados. Además, no solo se han de detectar las diferencias entre los bloques sino también se ha de corregir el bloque erróneo.
    b. Aplicar la tarea de sincronización de tiempos a todos los nodos.
    c. Aplicar la tarea del reseteo a uno o a todos los nodos.
3. ***NMR***. Basada en la replicación impar (mayor que 3) del nodo, con lo que el número de fallos detectados y corregidos puede ser mayor o igual que 2.
4. ***Redundancia selectiva***. Redundancia de tareas críticas. Las tareas críticas que están en un nodo se replican para cada nodo de la red, como ya se ha explicado anteriormente. Esta es una técnica para asegurar una correcta ejecución de las tareas gracias a la votación por mayoría.
5. ***Identificación y aviso del nodo fallido***, mediante la generación de un sistema que sea capaz no sólo de detectar y corregir un fallo (redundancia pasiva) sino también de informar sobre el elemento que está teniendo el fallo.
6. ***Endurecimiento selectivo del protocolo de comunicación***, introduciendo mejoras funcionales en el mismo.

En general, la evaluación de la robustez del sistema final alcanzada con estas técnicas se realiza en su aplicación final, con su funcionamiento en campo (*in-field*). Hoy en día, hay muy pocos *tests* de validación de robustez que se realicen mediante campañas de inyección de fallos en el sistema completo. Esto es debido a la dificultad

de observar los efectos de los fallos y de distinguirlos frente a los efectos de retrasos temporales o del ruido, con lo cual verificar que las técnicas de mitigación funcionan correctamente resulta prácticamente imposible.

Si se aumenta la observabilidad de cada elemento de la red, es posible obtener la información necesaria para verificar si el comportamiento del sistema con fallos presenta el nivel requerido de tolerancia a fallos. De hecho, esto ofrece las siguientes posibilidades:

- Facilita la evaluación de las técnicas de endurecimiento aplicadas. En concreto, con los experimentos de irradiación, la localización del fallo no se puede controlar. Al aumentar la observabilidad, el diseñador puede obtener la información sobre las partes del sistema afectadas por un fallo y sobre qué técnicas de endurecimiento han funcionado.
- Detecta puntos débiles de los componentes antes de aplicar las técnicas de endurecimiento. Así, se puede efectuar un proceso de endurecimiento eficiente, aplicando las técnicas de endurecimiento específicas.
- Informa sobre el efecto del fallo, aunque su efecto no haya llegado a las salidas. Esta información se puede utilizar durante la ejecución en campo del sistema, para prevenir la acumulación de los fallos y para la realización de un diagnóstico de los fallos permanentes, todo esto con el fin de mejorar las técnicas de endurecimiento aplicadas.

Por último, como las tareas críticas se replican y se aplica la votación por mayoría de sus resultados (endurecimiento colaborativo), los fallos aislados (localizados en los nodos de la red) se podrán detectar y corregir. Por lo tanto, en esta tesis se presenta un conjunto de técnicas y módulos de aumento de la observabilidad en los sistemas electrónicos digitales.

5.1.1. Verificación local

5.1.1.1. Informe interno del error.

Aquellos elementos en el nodo de la red que están ejecutando tareas globales y críticas se duplican y sus salidas se comparan mediante la técnica de redundancia *hardware* activa (*Duplication with comparison*, DWC) para detectar la presencia de fallos. En el caso de redundancia hardware, DWC se puede utilizar para la detección de SETs y SEUs en la lógica combinacional y secuencial. Además, debido a su sencilla implementación, existen varias herramientas CAD para FPGAs que aplican automáticamente esta técnica.

La estructura básica de DWC se muestra en la Figura 36. El circuito original se duplica completamente para formar dos idénticos bloques. Este circuito detecta cualquier diferencia en las salidas de los dos módulos. Si se detecta alguna diferencia, un *flag* de error se activa para indicar que los módulos no muestran comportamiento idéntico.

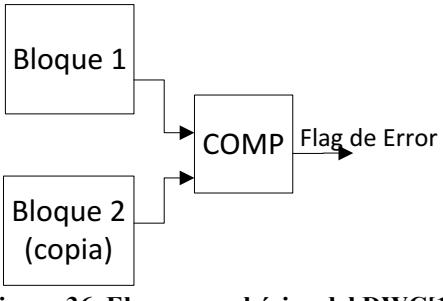


Figura 36. El esquema básico del DWC[109].

La novedad implementada en este método propuesto para una red distribuida, consiste en que la presencia de errores se acumula y se informa en forma *on-line* al nodo maestro del sistema. El esquema del DWC ampliado se muestra en la Figura 37. El nodo maestro recopila la información de todos los nodos del sistema y actúa de acuerdo con la aplicación programada.

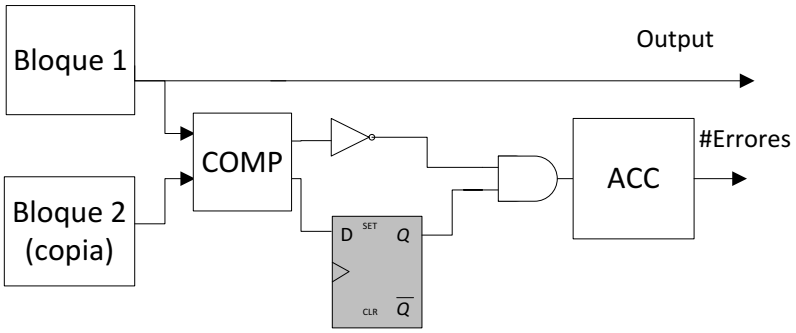


Figura 37. El esquema del DWC ampliado para un sistema distribuido [109].

5.1.1.2. Circuito generador de minoría *Minority checker*

Cuando varios elementos en un sistema distribuido están realizando la misma tarea crítica, es necesario conocer cuál de ellos está fallando, y procesar si hay varios bajo fallo en concurrencia. Con el fin de detectar el nodo que está fallando se aplica a las salidas de los elementos bajo análisis (nodos esclavos del sistema distribuido completo), un circuito de comprobación. Este circuito de comprobación tiene tantas salidas como bloques que se están analizando, y se activan todas aquellas que corresponden al bloque que no se comporta como la mayoría (bloque minoritario) [106]. El circuito permite conocer qué bloque está fallando, aunque en el sistema distribuido esté funcionando un sistema de votación por mayoría que mitiga y corrige los fallos simples de los nodos. De esta forma, el sistema en su conjunto continua funcionando correctamente, pero tiene constancia, en tiempo de operación, de qué bloques fallan y con qué frecuencia, para acometer las tareas de mantenimiento y reparación necesarias.

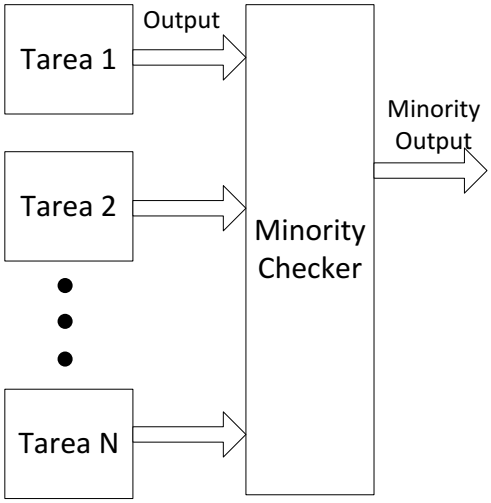


Figura 38. Circuito generador de minoría [109].

5.1.2. Verificación distribuida

5.1.2.1. Votador de mayoría con identificación del nodo que falla.

La redundancia N-modular (NMR) es una técnica que replica el bloque N veces y añade un bloque de votación por mayoría. Como medida de diagnóstico, el votador, además de producir una salida correcta, por mayoría, y corregir los nodos que están fallando, identifica la réplica con la tarea fallada. En el caso de un sistema distribuido con varios nodos que realizan tareas iguales, es preciso añadir esta funcionalidad. Por tanto, para el sistema distribuido complejo, el bloque NMR genera una salida de N-bits, donde cada bit es equivalente a 0 si la réplica correspondiente no ha fallado y es equivale a 1 en caso de error. La corrección del error se ha implementado en cada nodo de la red, pero la información sobre el nodo o nodos fallidos es útil para analizar la aparición de los fallos, sus efectos y realizar un mantenimiento *on-line* del sistema. Aunque el votador de mayoría podría implementarse en software, la solución hardware ha sido utilizada ya que proporciona más rapidez y eficiencia [107].

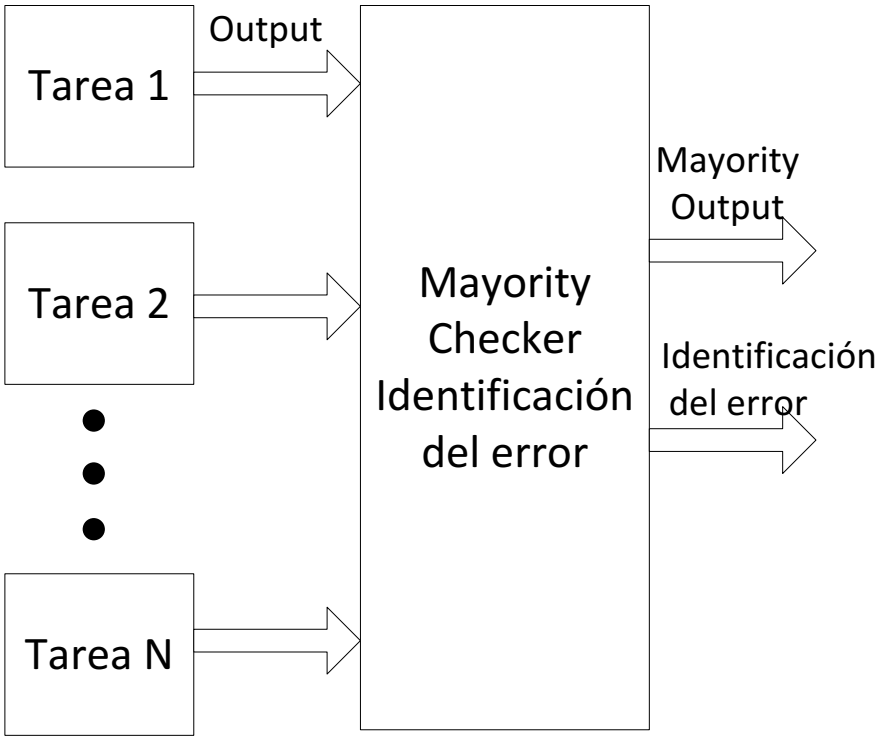


Figura 39. Circuito votador de mayoría [109].

5.1.2.2. Bus de comunicación.

En general, los protocolos de comunicación ya tienen incluidos un mecanismo para detectar los fallos que afectan al bus desde fuera. Los protocolos proporcionan cierta robustez en la transmisión de datos, pero la tolerancia a fallos no está garantizada en la lógica de control de la interfaz. Distinguir entre fallos que afectan a los módulos de comunicación y fallos en los nodos es una tarea importante en el proceso de evaluación de la robustez, con el fin de detectar los elementos que requieren soluciones de endurecimiento adicionales. Cuando el bus de comunicación utilizado incluye mecanismos de detección de errores, el nodo maestro debe utilizarlos para realizar la comprobación necesaria. De lo contrario, es necesario incluir un mecanismo en el maestro que sea capaz de diagnosticar si el fallo ha afectado al hardware de comunicación.

Combinación de la información obtenida			
Técnicas de Detección de Error			Determinación del fallo
Informe interno del error o *Minority checker*	Votador de mayoría	Bus de comunicación	
Sí	No	No	*Soft Error* en los nodos particulares. El error no se propaga al nodo maestro
	Sí	No	*Soft Error* en los nodos particulares. Se detecta por el endurecimiento colaborativo
	No	Sí	Error en la comunicación (retardos, ruidos)
No	Sí	No	*Soft Error* en el nodo maestro, en el Votador
	No	Sí	*Soft Error* en los módulos del controlador
	Sí	Sí	*Soft Error* en el nodo maestro y en el controlador

Tabla 27. Combinación y los efectos de las técnicas propuestas.

5.2. RESULTADOS EXPERIMENTALES.

A lo largo de la realización de esta tesis doctoral, se han realizado tres campañas de irradiación con un haz de protones, generado por un ciclotrón situado en el Centro Nacional de Aceleradores de la Universidad de Sevilla [103]. Este acelerador produce protones con energías de hasta 18MeV. El uso de estos aceleradores de baja energía está siendo muy recomendado en la actualidad para calificar los circuitos electrónicos digitales con dimensiones nanométricas. En la primera campaña, el objetivo principal era la caracterización de los dispositivos configurables para su uso en aplicaciones aeroespaciales; sin embargo, esta campaña y las dos posteriores han permitido la validación de las técnicas propuestas para la realización del *test* remoto y *on-line* de sistemas distribuidos, no sólo para su diseño y desarrollo, sino para la realización del *test* de campo.

En la primera campaña de irradiación han participado el Centro Nacional de Aceleradores (CNA), la Universidad Carlos III de Madrid (UC3M) y el Instituto Nacional de Técnica Aeroespacial (INTA), aportando este último el sistema real aeroespacial (satélite OPTOS) cuyos componentes era necesario caracterizar, para futuras misiones, y de los cuáles, por ser comerciales los fabricantes no proporcionaban datos sobre su robustez frente a la radiación ionizante. La segunda campaña se valió de la primera para validar los primeros bloques de *test on-line*, *minority checker,* aplicándolos a la caracterización de más dispositivos. Finalmente, en la tercera campaña se ha comprobado el funcionamiento del sistema distribuido completo bajo la radiación ionizante y, sobre todo, se ha comprobado la validez de los bloques propuestos para la detección de los fallos, la comprobación de las técnicas de mitigación y el *test on-line*.

Durante los *test* realizados en el CNA con protones de baja energía (<20 MeV) se han evaluado tres dispositivos programables: la Coolrunner de Xilinx y la Igloo de Microsemi y el microcontrolador Cortex-M3 de ARM de ST Microelectronics. Aunque el *test* realizado correspondió a *Single Event Effects* (SEE) y no al de Dosis Acumulada

(TID), y a pesar de que el valor de las energías utilizadas fue muy bajo se han de tener en cuenta los daños por dosis debidos a las altas fluencias de los protones.

En cada campaña, previamente a la irradiación se ha analizado la sensibilidad del diseño y se ha realizado un endurecimiento completo o parcial.

Las campañas de irradiación se realizaron sobre dispositivos reconfigurables basados en memorias Flash, FPGAs de Xilinx™ y de Microsemi®, que presentan una tasa de fallos baja para las energías y las partículas utilizadas. La elección de la FPGA no es un problema trivial, ya que resulta ser conveniente la caracterización de la sensibilidad de la tecnología antes de realizar el estudio del diseño que irá prototipado sobre ella. En este aspecto, es necesario comparar dispositivos de distintos fabricantes y de distintos tamaños y capacidades. Por último, aunque existan muchos trabajos publicados sobre la irradiación de circuitos digitales con distintos haces de partículas, la comunidad científica no ha enunciado una metodología de verificación completa de sistemas distribuidos digitales, mediante irradiación.

En esta tesis doctoral se estudian y se comprueban las distintas tecnologías con un bloque especifico, se verifican características de circuitos integrados sometidos a bajas energías de irradiación con protones, se realiza la campaña de irradiación del sistema distribuido completo con la tecnología elegida y finalmente se establece una metodología de verificación completa de un sistema distribuido sometido a la irradiación.

5.2.1. Campaña de irradiación 1

En esta campaña se aplicó el plan de pruebas que el INTA utilizaba en otras instalaciones europeas, permitiendo el ajuste del ciclotrón a los requisitos de uniformidad, flujo, fluencia y demás [108]. El sistema de pruebas utilizado fue proporcionado por el departamento de Observación de la Tierra del INTA, permitiendo el control remoto de los dispositivos de prueba, la parada y re-arranque ante detección de fallos tipo SEL, así como la detección e informe de los fallos tipo SEU y SEFI. Todo el diseño y el desarrollo del sistema digital a irradiar se realizó en el grupo de Diseño Microelectrónico y Aplicaciones de la Universidad Carlos III de Madrid, en colaboración con el citado departamento del INTA.

5.2.1.1. Implementación

El circuito de pruebas implementado es un multiplicador de 10 bits. Se ha dimensionado el circuito para maximizar el uso de los recursos de la CLPD, de modo que se han hecho visibles la mayor parte de los SEUs en la memoria de configuración. Se asumió que la probabilidad de la aparición de SEUs en la memoria de configuración es mucho mayor que la probabilidad de su aparición en los biestables, de modo que se ha optado por maximizar el uso de lógica combinacional en lugar del número de biestables.

El circuito multiplicador (*StageMult*), Figura 40 y Figura 41, se ha prototipado en la CPLD *CoolRunner-II* XC2C256. Este circuito recibe los datos y la condición de arranque desde otra CPLD (XC9752XL) que tiene como misión generar los estímulos para el multiplicador, Figura 42. Las salidas del multiplicador se envían al bloque interfaz que decide si ha aparecido un error durante la operación del multiplicador.

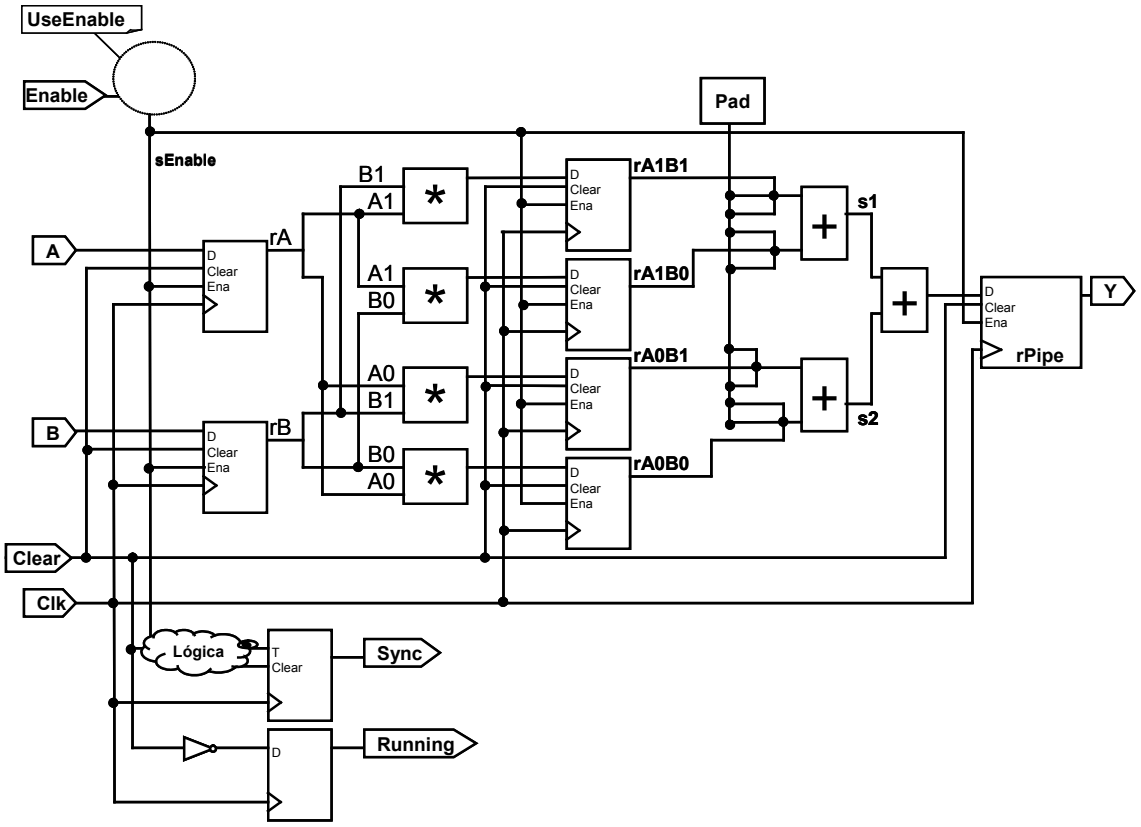


Figura 40. Diagrama de bloques del multiplicador *StageMult*

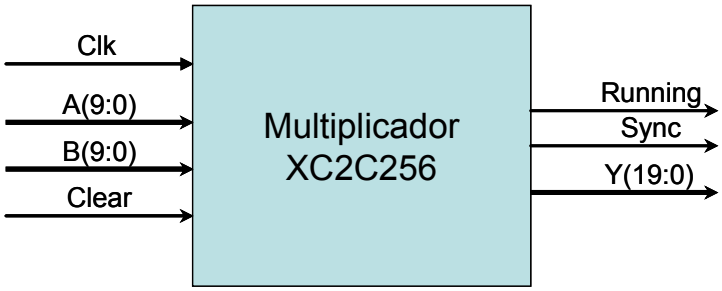


Figura 41. Entradas y salidas del multiplicador

El generador de estímulos es un circuito genérico, el cual a partir de las señales *clock*, *start* y stop genera las dos entradas pseudoaleatorias de 10 bits para el bloque multiplicador.

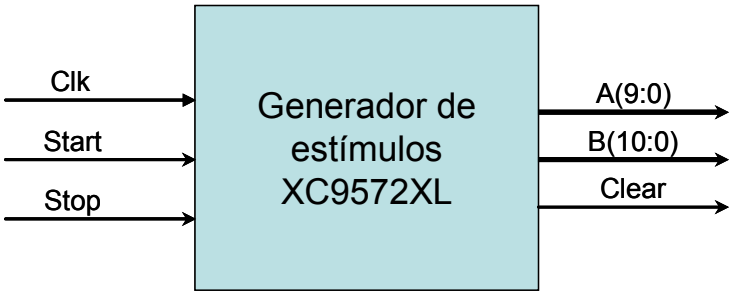


Figura 42. Entradas y salidas del generador de entradas al multiplicador

Endurecimiento del Circuito

El circuito multiplicador, *StageMult*, se ha analizado para su posible endurecimiento frente a los fallos SEU mediante las técnicas de redundancia. Las técnicas de redundancia consideradas han sido:

1. Técnicas de redundancia del hardware
    a. Una técnica pasiva, que corrige el fallo mediante un enmascaramiento del fallo.
    b. Una técnica activa, que solo detecta y no corrige, mediante la detección de la presencia de fallo.
2. Técnicas de redundancia de la información
    a. Una técnica que consiste en inserción de bits de paridad en todos los elementos de la memoria, susceptibles de sufrir *SEUs.*

Versión endurecida por TMR

El circuito multiplicador con redundancia hardware pasiva TMR se implementó a partir de la *netlist* sintetizada del circuito con la herramienta software. Esta técnica replica tres veces cada elemento de memoria del circuito y realiza una votación con una actualización de los valores contenidos en las tres réplicas. Los fallos provocados por *SEUs* son enmascarados y corregidos.

Esta técnica supone un incremento de área considerable, de forma que no puede implementarse en una CR2-256 (en una CR2-512 sí hay recursos suficientes).

Versión endurecida por bit de paridad

Para la generación del circuito multiplicador con el bit de paridad se ha desarrollado un paquete de funciones de generación y comprobación de la paridad que se han incluido en la versión con la paridad del multiplicador. Esta técnica supone un pequeño incremento del área y permite la detección de errores simples en todos los elementos de la memoria del circuito así como en los bits de paridad. La generación de un bit de error como salida adicional del circuito permite conocer durante la operación normal del circuito la presencia de dichos fallos.

5.2.1.2. Campaña de irradiación: Resultados experimentales

Emulación

Se ha realizado un estudio del efecto que tendrían los fallos SEU en la funcionalidad del diseño; tanto en el circuito sin endurecer como en sus versiones endurecidas. Para ello se ha aplicado un método de inyección de fallos transitorios basado en la emulación hardware con FPGA siendo el modelo del fallo adoptado el *bit-flip*.

Los fallos transitorios vienen determinados por el elemento de la memoria al que afectan y el instante de tiempo en el que ocurren. Las campañas de fallos realizadas consisten en la inyección de todos los posibles fallos simples, esto es, se inyectan fallos en todas las posibles localizaciones (biestables) y para todos los instantes posibles de tiempo (cada ciclo de reloj del banco de pruebas). De esta forma se obtiene el diccionario completo de fallos.

El banco de pruebas considerado consta de 1.048.590 ciclos de reloj (contiene todos los valores posibles para los operandos de entrada al multiplicador, $2^{10} \cdot 2^{10} =$ 1.048.576). Durante los primeros cuatro ciclos la señal *clear* está activada y, por lo tanto, el multiplicador está parado. Se considera que los fallos inyectados a partir del

ciclo de reloj número 1.048.580, son equivalentes a otros ya inyectados puesto que la ejecución se repite. Por lo tanto, el número de posibles fallos es C*F, donde C= 1.048.580 y F es el número de biestables del circuito bajo prueba (82 para el circuito sin endurecer).

Los efectos observados se clasificaron en las siguientes categorías: Silencioso (*silent*) si el efecto del fallo desaparece durante la ejecución del banco de pruebas, Avería (*failure*) en el caso de que se produzca un error en alguna salida del circuito, Latente (*latent*) si las salidas no se han visto afectadas pero el fallo permanece almacenado en el circuito al finalizar el banco de pruebas. Además, en el caso de circuitos que disponen de algún mecanismo de detección de errores, como en el caso de usar código de paridad o la técnica DWC, la clasificación de los fallos también refleja el número de los que se han detectado y los que no.

El diccionario obtenido de fallos sin endurecer es el siguiente:

# Silenciosos	# Latentes	# Averías	# Total de fallos inyectados
20.680 (0,02%)	0 (0%)	85.962.880 (99,98%)	85.983.560

Tabla 28. Diccionario de fallos del circuito sin endurecer

Debido a que el diseño bajo estudio consistió en un circuito aritmético segmentado, la mayoría (99,98%) de los fallos inyectados causaron efectos en las salidas del circuito. Los fallos silenciosos se producían cuando el fallo se insertaba en un operando y el otro valía 0 o cuando se inyectaba en los 5 primeros ciclos durante los cuales ambos operandos valían 0. Podemos concluir que, la observabilidad de los biestables de este circuito es muy alta y esto permite una detección de *SEUs* muy buena.

El diccionario de fallos obtenido para el circuito endurecido con el TMR es el siguiente:

# Silenciosos	# Latentes	# Averías	# Total de fallos inyectados
85.983.560 (100%)	0 (0%)	0 (0%)	85.983.560

Tabla 29. Diccionario de fallos del circuito con TMR en todos los biestables

Este circuito presentaba el doble de biestables que el original, sin embargo los fallos que afectaban a cualquiera de los elementos redundantes eran equivalentes, por lo que el número de fallos estudiados fue el mismo que en la campaña de fallos realizada al circuito original.

Al aplicar TMR a todos los biestables del circuito se enmascaran todos los fallos simples y aquellos múltiples que no afectan a los dos biestables redundantes. En este caso si el sistema interfaz detectase algún fallo, las causas podrían ser alguna de las siguientes:

- Un SEU en la memoria de configuración.
- Un error múltiple, MBU (*Multiple Bit Upset*) o SET que generen MBU, que afecte al menos a dos de los tres biestables que forman cada unidad de TMR.

El diccionario de fallos obtenidos con el endurecimiento con la paridad es el siguiente:

# Silenciosos			# Latentes			# Averías			# Total inyectados
ND	D	Total	ND	D	Total	ND	D	Total	
0 (0%)	7.360.740 (7,89%)	7.360.740 (7,89%)	0 (0%)	0 (0%)	0 (0%)	2.097.160 (2,25%)	83.865.720 (89,87%)	85.962.880 (92,11%)	93.323.620

Tabla 30. Diccionario de fallos del circuito con código de paridad

En este caso, el circuito tiene biestables adicionales encargados de almacenar el bit de paridad por lo que el número de posibles fallos simples es mayor. En la Tabla 30 se pueden ver el porcentaje de fallos que han sido detectados (D) a partir del código de paridad y los no detectados (ND). Puesto que este método de endurecimiento no enmascara fallos los efectos que causan son los mismos que en el caso del circuito sin endurecer, a excepción de los fallos que se inyectan en los bits de paridad. Si un fallo afecta al bit de paridad se clasificará como silencioso y detectado lo cual explica por qué ahora el porcentaje total de fallos silenciosos es mayor. Los fallos que en la versión sin endurecer eran silenciosos ahora se detectarían originándose un error de paridad. Con respecto a las averías, el 97,56% eran detectadas.

En resumen, utilizando el código de paridad el sistema interfaz sería capaz de detectar los siguientes fallos:

- cualquier error del tipo SEU que se produzca en el sistema, bien comparando las salidas o bien por la señal que indica error en la paridad.
- errores múltiples que no afecten a la misma palabra.

Radiación

Hay un circuito del control encargado de la interfaz con el usuario, el control del circuito generador de los estímulos, la comprobación de los resultados de la multiplicación y de la recopilación de los resultados.

La interfaz con el usuario se realiza por medio de comandos, que se introducen en el PC (hyperterminal) y se transmiten mediante el protocolo RS232 al circuito de control. El circuito de control devuelve la información relativa a la ejecución al PC (*hyperterminal*), y esta información se almacena en unos ficheros de texto.

El circuito de control está basado en el microprocesador Microblaze de Xilinx, y se ha diseñado utilizando la herramienta Xilinx EDK 7.1. Se ha diseñado un periférico llamado *Radiation Watcher* (*radwatch*) que se comunica con Microblaze y que desempeña las funciones de control, comprobación de los resultados y recopilación de los resultados.

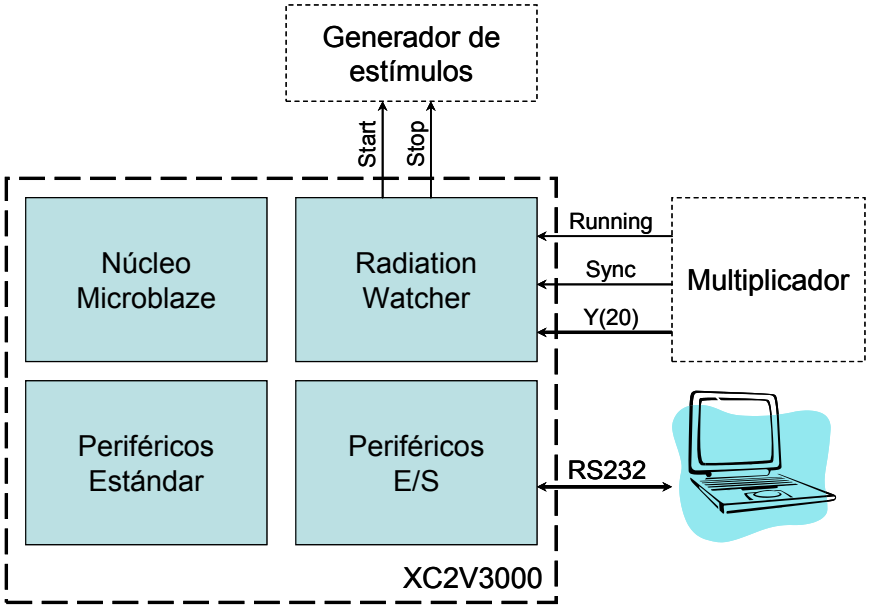


Figura 43. El esquema del sistema a testear (Diseñado por el INTA y DMA-UC3M).

5.2.1.3. Resultados

En esta campaña se han realizado 26 *tests* de irradiación con protones en el ciclotrón del Centro Nacional de Aceleradores de la Universidad de Sevilla. Se han ejecutado 13 *tests* sobre los dispositivos CoolRunner-II de Xilinx™. La tecnología utilizada es una CPLD CMOS de 0,18 micras, estándar comercial. Los dispositivos no estaban desencapsulados.

Los primeros 8 *tests* se realizaron para calificar la tecnología del dispositivo. La incidencia del haz se ha hecho sobre el dispositivo conectado a la alimentación, con el hardware configurado (como un multiplicador de dos números de 10 bits), pero sin activar la operación (la salida del circuito siempre es cero). Las comprobaciones que se realizaron durante la prueba fueron en las dos memorias de configuración, la memoria SRAM y la memoria Flash.

RUN	*MeV*	*Krad*	*SEUs*	*SELs*
1 (CR 2)	10,0 (aprox.)	30 aprox.	0	0
2 (CR 2)	10,0 (aprox.)	20 aprox. (acumulado 50)	0	0
3 (CR 2)	10,0 (aprox.)	30 aprox. (acumulado 80)	0	0
4 (CR 3)	14,7	30 aprox.	0	0
5 (CR 3)	14,7	20 aprox. (acumulado 50)	0	0
6 (CR 3)	14,7	30 aprox. (acumulado 80)	0	0
7 (CR 1)	17,3	30 aprox.	103	0
8 (CR 1)	17,3	20 aprox. (acumulado 50)	800 (>40)	0

Tabla 31. Los primeros 8 RUNs de irradiación.

En este experimento los fallos reportados por la interfaz JTAG que hacía la lectura de la memoria de configuración (Readback) de la CoolRunner-II pasaron de 40 a 400 → 800 y a continuación oscilaban aunque no hubiera haz de protones aplicado. Para el cálculo de la estimación de la tasa de SEUs se ha considerado un dispositivo (CR1) que a 14,7 MeV no presentaba errores y a 17,3 MeV (RUN 7) se obtuvieron 103 SEUs.

En el RUN 8 el dispositivo falló generando numerosos errores y un aumento de consumo por lo que se deduce que la degradación y finalmente la rotura del dispositivo se debió a un efecto de dosis acumulada que correspondía a un valor comprendido entre los de 30 krads(Si) (RUN 7) y los 51,54 krads(Si) (RUN 8).

En la Figura 44 se muestra el aumento del consumo recogido con el osciloscopio durante los RUN 7 y 8. Los aparatos electrónicos de medida, igual que otros dispositivos para los *tests* han sido proporcionados por INTA [104].

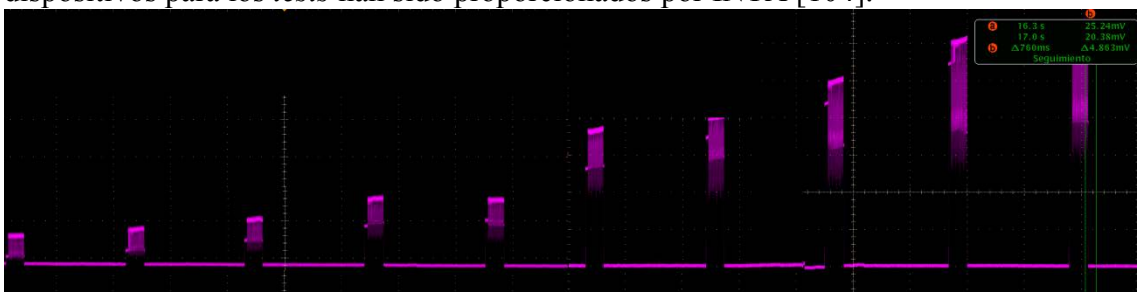


Figura 44. Aumento del consumo durante el test de irradiación.

Como conclusión sobre estos *tests* estáticos aplicados al dispositivo CoolRunner-II se puede apreciar que la tecnología no es sensible a protones con energías inferiores a los 17 MeV. Para energías superiores a este valor, sí se observa una notable sensibilidad en la memoria SRAM de configuración. La memoria Flash no ha presentado ningún fallo en los *test* estáticos.

Los cinco últimos *tests* realizados sobre la CoolRunner-II de Xilinx™ se han aplicado sobre el dispositivo con el hardware configurado (como un multiplicador de dos números de 10 bits) y con la operación activada. De forma continua durante los *tests* se han estado comprobando los resultados de la operación y anotando los SEFIs. En estos *tests* se comprobó no sólo la sensibilidad de la memoria SRAM de configuración sino también la de los elementos de memoria que están conformando el circuito en sí.

RUN	*MeV*	*Krad*	*SEFIs*	*SELs*
9 (CR 4)	17,3	30 aprox.	23	2
10 (CR 5)	14,7	29,77	0	0
11 (CR 5)	17,3	30 aprox.	9	0
12 (CR 6)	17,3	30 aprox.	25	0
13 (CR 6)	17,3	30 aprox. (60 krad acumulados)	14	1

Tabla 32. Los 5 últimos RUNs sobre CRII.

Para la energía de 14,7 MeV no se han observado ningún SEFI ni SEL, lo que se traduce en un aumento excesivo de la corriente demandada a la fuente de alimentación y en una destrucción del dispositivo si esta corriente no se corta rápidamente. Con la energía de 17,3 MeV se han observado decenas de SEFIs y algún SEL. Esto prueba que la sensibilidad de este dispositivo (todos los elementos de la memoria SRAM) a protones con energías superiores a 17 MeV se hace notar.

Los siguientes 7 *tests* se han aplicado al dispositivo Igloo de Microsemi™. Las FPGAs Igloo utilizan transistores CMOS de 130-nm. Este dispositivo presenta una memoria Flash de configuración y la única memoria SRAM que presenta esta en los elementos secuenciales del circuito configurado.

Todos los *tests* realizados fueron dinámicos, aplicados sobre el dispositivo con el hardware configurado (como un multiplicador de dos números de 10 bits) y con la operación activada. Una vez realizado cada *test* se ha comprobado la integridad de la memoria Flash de configuración.

RUN	*krad*	*SEUs*	*SELs*	*Test Flash*
14 (IG 1)	30 aprox.	0	0	OK
15 (IG 1)	30 aprox. (acumulado 60)	1	0	OK
16 (IG 1)	30 aprox. (acumulado 90)	0	0	Fallos en el cto JTAG
17 (IG 3)	30 aprox.	0	Fallos (50%)	0
18 (IG 3)	30 aprox. (acumulado 60)	0	0	Fallos en Flash sobre el 70%
19 (IG 3)	30 aprox. (acumulado 90)	0	0	No se reconoce cadena JTAG
20 (IG 3)	30 aprox. (acumulado 120)	Fallo permanente	0	Fallos en el cto JTAG

Tabla 33. Los 7 *tests* sobre el dispositivo Igloo.

Como conclusión a estos *tests* sobre el dispositivo Igloo se puede apreciar que la tecnología es menos sensible a los protones con energías hasta 17 MeV, que el dispositivo CoolRunner-II, pero a energías superiores a 17 MeV y mucha dosis acumulada se provoca la destrucción del circuito de configuración del dispositivo, aunque la memoria Flash sigue configurada hasta el valor de unos 60 krad aproximadamente.

Los siguientes seis *tests* se han realizado sobre un dispositivo comercial, el microprocesador Cortex-M3 de ST™. Fueron cuatro *tests* estáticos en los cuales se ha comprobado la integridad de la memoria RAM del dispositivo, y dos *tests* dinámicos en los cuales, se han comprobado el funcionamiento del microprocesador y de la interfaz de comunicación.

RUN	*MeV*	*Krad*	*SEUs/ SEFIs*	*SELs*
21 (ST 1) Estático	17.3	30 aprox.	18	0
22 (ST 1) Estático	17.3	30 aprox. (60 krad acumulados)	40	0
23 (ST 2) Dinámico	17.3	30 aprox.	2	0
24 (ST 2) Dinámico	17.3	30 aprox. (60 krad acumulados)	3	0
25 (ST3) Estático	14.7	30 aprox.	6	0
26 (ST3) Estático	14.7	30 aprox. (60 krad acumulados)	14	0

Tabla 34. Los 6 *test* sobre el microprocesador Cortex-M3.

Se ha observado una notable sensibilidad de este dispositivo frente a los protones, incluso a energías cercanas a los 14 MeV.

Análisis de resultados de la FPGA CoolRunner™ de Xilinx®

Para la determinación de la sensibilidad a SEUs de la CoolRunner se testearon 3 dispositivos: CR2, CR3 y CR1 a tres energías diferentes con valores del error mostrados en la siguiente tabla. A 9.9 MeV y 17.7 MeV no se detectaron errores ni tampoco aumentos en el consumo debidos a la dosis acumulada hasta 80 krads (Si). Teniendo en cuenta que el espesor del encapsulado de la Coolrunner es de 2mm y analizando el rango de penetración en Silicio utilizando SRIM (Stopping and Range of Ions in Matter) para estas energías, se deduce que estos protones no llegan a la zona sensible del dispositivo y por lo tanto no producen errores. El rango de penetración para protones de 17.3MeV es mayor que 2 mm y por lo tanto ésta es la única energía con capacidad de provocar SEUs.

Energy (MeV)	**Error (keV)**	**Rango en Silicio**
17,3	245	1,05 mm
14,7	332	2,10 mm
9,9	540	2,82 mm

Tabla 35. Los rangos de penetración en el Silicio para los protones.

Análisis de resultados de la Igloo

Se irradiaron 3 dispositivos mediante un *test* dinámico utilizando la energía más alta disponible en la instalación. Debido a las bajas energías con las que se ha realizado el *test* en el CNA sólo se puede predecir que la energía umbral, es decir, aquella a partir de la cual hay fallos para una fluencia superior a 10^{11} protones /cm^2 es superior a 17,3

MeV y por tanto este dispositivo resulta ser menos sensible que la CoolRunner. Para poder determinar la sensibilidad y por tanto la tasa de fallos habría que realizar un *test* con protones a energías mayores.

Atendiendo a la resistencia a la dosis total, la IG-1 no sufrió degradación hasta el valor de 61,5 krads (Si). La IG-3 se irradió hasta que se rompiera el dispositivo, encontrándose que a partir del valor de 100krad (Si) los fallos en el JTAG eran del orden del 70%.

5.2.2. Campaña de irradiación 2

En la segunda campaña de irradiación el propósito fue comparar las distintas tecnologías bajo irradiación con el mismo circuito bajo *test*, utilizando el bloque *Minority Checker* propuesto en esta tesis doctoral. En función de la tecnología elegida se modificaban algunas formas de testear el circuito.

En esta campaña, tres dispositivos CoolRunner II ™ de Xilinx® y un dispositivo Igloo® de Microsemi® han sido elegidos para ser comparados. Estos dispositivos están bajo consideración para su uso en la segunda versión de OPTOS CubeSat, desarrollado por el INTA [65]. Para que los dispositivos de las dos tecnologías puedan ser comparados, se ha requerido un análisis previo a la campaña de irradiación con el objetivo de la obtención de medidas comparables de ambos fabricantes, los cuales presentan diferentes tecnologías, memorias de configuración y arquitecturas internas, así como distintos recursos disponibles.

La tabla 36 presenta las características principales de los dispositivos de Xilinx® y Microsemi®.

Cool Runner II de Xilinx

Los dispositivos CPLD CoolRunner-II (CRII) de la empresa Xilinx no están diseñados para funcionar en entornos de alta radiación, pero sus características especiales los hacen muy adecuados para el uso en el sector espacial. Presentan un consumo de energía muy bajo (28,8 mW) y junto con los factores de encapsulado pequeño hacen de CRII una herramienta valiosa para hacer frente a los sistemas de baterías de alimentación y múltiples sensores redundantes. Esta tecnología está basada en memoria no volátil y presenta transistores CMOS de 180 nm.

La memoria no volátil (Flash) almacena la información de configuración que puede ser descargada en la memoria de configuración volátil (SRAM) cada vez que el dispositivo se encienda o se reinicie. Se han elegido dispositivos de 32, 256 y 512 macrocéldas para compararse con la tecnología de Microsemi®.

Igloo de Microsemi (Antiguamente Actel)

La serie Microsemi® Igloo® corresponde a las FPGAs de baja potencia y debido a esta característica resultan ser muy adecuadas para los propósitos de la ingeniería espacial. La familia Igloo® presenta una FPGA de gran densidad, de bajo consumo y con capacidad extra de pines, lo que la convierte en un dispositivo valioso para pequeños satélites con presupuestos no elevados, de baja potencia y altos requerimientos computacionales. La FPGA Igloo está basada en la tecnología Flash reprogramable. Las FPGAs basadas en esta tecnología requieren un menor consumo de energía que las FPGAs basadas en SRAM y además la serie Igloo proporciona un modo estático de baja potencia, llamado modo Flash*Freeze. Las FPGAs Igloo utilizan transistores CMOS de 130-nm con siete capas de metal.

	CRII CPLD, Xilinx			Igloo FPGA Microsemi
Dispositivo	XC2C32 QFG32	XC2C256 7TQ144	XC2C512 7PQG208	M1AGL1000 FGG484
CMOS (nm)	180	180	180	130
Length x Width (mm)	5x5	20X20	28x28	23x23
# pins Total/User	32/21	144/118	208/173	484/300
# of Macrocells	32	256	512	24,576
# bits in *StageMult*	4	10	4	10
# of instances of *StageMult*	1	1	8	25

Tabla 36. Características principales de los dispositivos de Xilinx y Microsemi.

5.2.2.1. Implementación

En primer lugar, con el fin de evaluar la efectividad del bloque *Minority Checker*, se compararon dos dispositivos del mismo fabricante: CRII-32 y CRII-512. El multiplicador de 4 bits *StageMult* se ha prototipado en una CRII-32 con una ocupación de área total de 85%. Por otro lado, en una CRII-512 el mismo multiplicador sólo ocupa el 6% del area. Por lo tanto, para realizar una comparación equivalente se prototipa en una CRII-512 un total de 8 unidades del circuito *StageMult* de 4 bits y un bloque que calcula la minoría, dando como resultado un 89% de ocupación del área. Cabe destacar que el comprobador de minoría ocupa un 29% de los recursos del citado dispositivo.

Dispositivo	Recursos Usados / Total	Lógica combinacional Usados / Total	Pines Usados / Total	Registros Usados / Total
CRII-XC2C32 *SM*(4-bit)	27/32 (85%)	86/112 (77%)	20/21 (95%)	18/32 (57%)
CRII XC2C512 *SM* (4-bit)	27/512 (6%)	9/512 (2%)	20/173 (12%)	18/512 (4%)

Tabla 37. Resultados de la implementación del circuito *StageMult* de 4 bits para distintos dispositivos Xilinx.

Dispositivo	Recursos Usados / Total	*Total*	*Minority Checker*		
		Lógica Comb. Usados/ Total	Registros Usados / Total	Lógica Comb. Usados / Total	Registros Usados / Total
CRII XC2C512 *SM*N (4-bit) N=8 units	453/512 (89%)	309/512 (60%)	144/512 (29%)	150/512 (29%)	0/512 (0%)

Tabla 38. Ocupación de área de *StageMult*N &Minority Checker para *CoolRunner* II-512 de Xilinx

En segundo lugar, teniendo en cuenta la posibilidad de comparar distintas tecnologías, se elige el fabricante Microsemi de las FPGAs. Los dispositivos seleccionados de estos dos fabricantes deben de implementar un circuito equivalente para poder compararlas, aunque internamente sus arquitecturas sean diferentes.

Se ha seleccionado el dispositivo CoolRunner CR-II-256 de Xilinx, para compararlo al dispositivo Igloo de Microsemi.

El dispositivo CRII-256 es un dispositivo más pequeño en términos de área que el dispositivo Igloo. El circuito multiplicador *StageMult* (SM) de 10 bits se ha prototipado en el dispositivo CR-II-256 con una ocupación del área de 77%, mientras que este mismo multiplicador en el dispositivo M1AGL1000 ocupó solo un 2,71%.

Dispositivo	Macrocells Usados / Total	Lógica Comb. Usados / Total	Pines Usados / Total	Registros Usados / Total
CRII-XC2C256 *SM*(10-bit)	196/256 (77%)	805 /896 (90%)	44 /118 (37%)	82 /256 (32%)
Igloo M1AGL 1000 *SM*(10-bit)	667/24,576 (2.71%)	585 / 24,576 (2.38%)	44 / 300 (14.67%)	82 / 24,576 (0.33%)

Tabla 39. Resultados de la implementación del circuito *StageMult* de 10 bits para dispositivos Xilinx y Microsemi.

Dispositivo	Recursos Usados / Total	*Total*		*Minority Checker*	
		Lógica Comb. Usados / Total	Registros Usados / Total	Lógica Comb. Usados / Total	Registros Usados / Total
Igloo M1AGL 1000 *SM*N(10-bit) N=25 unidades	22,410 / 24,576 (91%)	20,488 / 24,576 (83.36%)	1,922 / 24,576 (7.8%)	4,180/24,576 (17%)	0/24,576 (0%)

Tabla 40. Ocupación de área de *StageMult*N &Minority Checker para dispositivo Igloo

La Tabla 39 muestra el área de un multiplicador *StageMult* de 10 bits para los dispositivos CRII-XC2C256 y M1AGL1000. Para CRII-XC2C256 la ocupación correspondió a un total de 90% de puertas lógicos combinacionales y sólo a un 32% de registros de memoria. El diseño del circuito se ha implementado de la manera anterior debido a que el principal objetivo del *test* era ver los efectos de los SEUs en la memoria de configuración, y su propagación a las salidas del circuito. Para la FPGA Igloo de Microsemi el mismo circuito requirió un 3% menos de recursos en términos de lógica combinacional. Replicando el circuito 25 veces y añadiendo el *minority checker* en la salida de los bloques, se ha conseguido una ocupación del 91% de la FPGA de Microsemi. En este caso, el corrector de minoría llegó a ocupar el 17% de los recursos de lógica combinacional. Este bloque esta implementado completamente de manera combinacional, y cada SEU que afecte a los bits de configuración se propagará a la salida del dispositivo.

5.2.2.2. Experimentos

Inyección de fallos transitorios mediante Emulación *Hardware*

El análisis de la sensibilidad ante los SEUs se ha realizado mediante cuatro campañas de inyección de fallos a través de la emulación de hardware. Dos diseños han sido evaluados dos veces: *StageMult* y *StageMult*N, la primera vez en los dispositivos de Xilinx y, en segundo lugar, en los dispositivos de Xilinx y Microsemi. El objetivo de estos experimentos consistió en comprobar la eficacia del *minority checker* con respecto a la propagación de los errores en los distintos circuitos, cuando se prototipan muchas réplicas del mismo circuito. Además, el experimento permite ver la efectividad del

bloque *minority checker* para cualquier número de réplicas y cualquier anchura de los datos comparados.

La emulación se realiza de tal forma que cada elemento de memoria recibe un SEU por cada ciclo de reloj de un *workload* funcional. La campaña de inyección de fallos se ha realizado mediante el uso de la plataforma de emulación autónoma de DMA-UC3M.

El sistema de inyección de fallos se realizó en una plataforma de evaluación XC5VLX110T de Xilinx. El tamaño y el tipo de la FPGA utilizado para la emulación de hardware no resulta ser relevante para los resultados experimentales.

En la emulación autónoma, los fallos inyectados se clasifican de acuerdo con el efecto producido en el comportamiento del circuito. Las categorías consideradas son las siguientes. Cuando el efecto de un error se propaga a las salidas, la clasificación de fallo se establece como *Avería*. Cuando el efecto del fallo ha desaparecido por completo dentro del circuito, el fallo es clasificado como fallo *Silencioso*. Cuando el efecto del fallo permanece en algunos elementos de la memoria del circuito, el fallo se clasifica como *Latente*.

En el circuito *StageMult* de 10 bits se ha inyectado un fallo en cada flip-flop (109 flip-flops) en cada ciclo de reloj de la carga de trabajo (1.572.954 ciclos de reloj). Esto corresponde a un total de 171.451.986 fallos inyectados. En la Tabla 41 se muestra la clasificación de los fallos detectados. Como resultado global, alrededor del 31,28% de los fallos fueron *Silenciosos* y el 0% *Latentes*. La tasa de *Averías* fue de 68,72%.

En el circuito *StageMultN* de 25 multiplicadores de 10 bits se ha inyectado un fallo en cada circuito flip-flop (2.300 flip-flops) en cada ciclo de reloj de la carga de trabajo (1.572.954 ciclos de reloj). El número total de fallos inyectados fue 3.617.794.200. Los resultados fueron equivalentes al circuito *StageMult*, tal y como se esperaba.

	StageMult	*StageMultN*	*StageMult*	*StageMultN*
	1 SM de 10- bit	*25 SM de 10-bit*	*1 SM de 4-bit*	*8 SM de 4-bit*
	#Fallos %	# Fallos %	# Fallos %	# Fallos %
Silenciosos	53.406.088 31,15%	1.131.446.650 31,27%	3.182 28,51%	32.734 37,44%
Averías	118.045.811 68,85%	2.486.345.500 68,73%	7.964 71,36%	54.550 62,40%
Latentes	87 0,00%	2.050 0,00%	14 0,13%	136 0,16%
Total	171.451.986 100,00%	3.617.794.200 100,00%	11.160 100,00%	87.420 100,00%

Tabla 41. Clasificación de la campaña de inyección de fallos

Las columnas 4 y 5 de la Tabla 41 (comenzando por el lado izquierdo) muestran la clasificación de los fallos obtenidos para los circuitos *StageMult* de 4 bits y *StageMult* de N bits (donde N=25). En *StageMult* los fallos han sido inyectados en 24 flip-flops en cada ciclo de reloj de la carga de trabajo (465 ciclos de reloj). El número total de fallos inyectados fue $465 \cdot 24 = 11.160$. En *StageMultN* los fallos han sido inyectados en 188 flip-flops en cada ciclo de reloj de la carga de trabajo (465 ciclos de reloj). El número total de fallos inyectados fue 87.420. Los resultados no fueron totalmente equivalentes, como en los circuitos con multiplicador de 10 bits, debido probablemente a las diferencias en la ocupación de área.

Multiple Event Upsets

Un subconjunto de fallos dobles fue inyectado en los dispositivos con multiplicadores de 10 bits, Tabla 42. Estos fallos fueron inyectados en cada par de flip-flops en cada ciclo de reloj de la carga de trabajo. La aparición de los fallos dobles provoca una reducción de los fallos silenciosos. En ambos circuitos, el efecto de estos fallos fue similar.

	StageMult		*StageMultN*	
	#Faults	%	#Faults	%
Silencioso	42.989.215	25,07%	1.106.132.184	30,57%
Avería	128.462.687	74,93%	2.511.659.966	69,43%
Latente	84	0,00%	2.050	0,00%
Total	171.451.986	100,00%	3.617.794.200	100,00%

Tabla 42. Resultados de la clasificación de fallos múltiples

Radiación

Durante los *tests* de radiación han sido probados dos diferentes tipos de CRII, CoolRunner XC2C32 - pequeña (CRS) y CoolRunner XC2C512 - grande (CRL), así como el dispositivo Igloo M1AGL1000. Los dispositivos bajo *test* fueron irradiados con protones a 17,3 MeV, 16.1 MeV y 14,7 MeV. Se han realizado las pruebas tanto dinámicas como estáticas.

El propósito de la prueba dinámica es comprobar la sensibilidad del dispositivo ante los SEUs, cuando se ejecuta una aplicación en la que se analiza la memoria de configuración SRAM. Sólo una parte de los bits de configuración se utiliza para una aplicación determinada. Por lo tanto, los resultados de esta prueba de radiación serán más realistas que los de la prueba estática. La aplicación que se ejecuta en el dispositivo corresponde a un multiplicador pipeline con dos operadores de 4 bits y un resultado de 8 bits. El multiplicador se ha prototipado en el dispositivo de tal forma que utilice casi todos los recursos disponibles dentro del dispositivo probado. Durante la ejecución, se realiza un sistema de control de los resultados de multiplicación con el fin de detectar cualquier error. Cada vez que se detecta un SEFI, el PC almacena los datos y resetea el dispositivo para continuar. Para esta prueba se realizó un sistema de monitorización desde el PC que comprobaba de forma periódica la memoria del dispositivo, mientras que el circuito estaba bajo radiación. También se realizó un *readback* continuo durante los experimentos de irradiación con el fin de detectar cualquier cambio en la memoria Flash o en las dos memorias SRAM. El *test* de radiación con el dispositivo apagado también se llevó a cabo con el fin de comprobar la memoria flash en este modo.

A la hora de realizar las pruebas de radiación con protones de baja energía, es necesario hacer un análisis exhaustivo de la pérdida de energía al irradiar, con el fin de poder calcular la dosis total absorbida por el dispositivo. Sin embargo, no resulta fácil de obtener todos los detalles sobre la composición y estructura de los dispositivos comerciales. Los dispositivos están contenidos en un encapsulado de resina epoxi y tienen una microestructura compleja. Con el fin de obtener una estimación aproximada de la dosis ionizante absorbida, en nuestros experimentos se ha decidido suponer un modelo simple sobre la base de un sistema de capas múltiples: resina epoxi / Al / Si. En

función del dispositivo, se introdujeron diferentes espesores (estimados a partir del corte de algunos dispositivos desechados) en el código SRIM2008[5].

5.2.2.3. Resultados

Los experimentos de irradiación permiten estudiar los efectos de los protones de baja energía en los dispositivos electrónicos, con el fin de comprobar la validez de dichos dispositivos para su funcionamiento en entornos de radiación, y para verificar los procedimientos de las pruebas de radiación.

En estos experimentos diferentes tipos de dispositivos programables han sido probados, los dispositivos CPLD CoolRunner-II basados en SRAM, con diferente capacidad y encapsulado, y una FPGA Igloo basada en la tecnología Flash. En cuanto a la sensibilidad frente a las SEEs, los resultados muestran una sensibilidad muy baja para los dispositivos basados en Flash y una sensibilidad baja para los dispositivos basados en SRAM. Otros experimentos con protones de energía más alta se deberían de realizar con el fin de completar la caracterización de estos dispositivos.

Un resumen de los resultados obtenidos se muestra en la Tabla 43. En la tabla se ve la energía incidente de protones (E_I) en la superficie y la energía del haz (E_R) cuando llega a la parte activa de los dispositivos. Para el cálculo de la dosis absorbida en krad en este intervalo de energías, la energía depositada en el dado Si (E_D) se debe de tener en cuenta ya que el valor LET (Si) no debe ser considerado como constante. En los dispositivos más grandes, el haz está totalmente parado en el área activa. Su encapsulado es tan grueso que los protones de energías bajas no pasan a través del silicio, por ejemplo, los protones de 3.1 MeV solo son capaces de atravesar menos de 100 micras. Por lo tanto, al igual que la dosis absorbida es máxima, los efectos secundarios tienen un papel muy importante.

E_I (MeV)	E_R in Si (MeV)	E_D (MeV/p)	Fluence/run (p/cm^2)	krad/Run	SEFI/Run	SEUs /Run
CoolRunner LARGE (CRL)						
17.3	6.4	6.4	8.50E+10	94	4	103
16.1	3.1	3.1	8.50E+10	45	0	0
14.7	—	—	7.50E+10	—	0	0
CoolRunner SMALL (CRS)						
17.3	12.6	2.2	8.50E+10	43	6	8
16.1	11.0	2.5	8.50E+10	49	2	3
14.7	9.1	3	7.50E+10	52	0	1
IGLOO						
17.3	14.7	2.2	8.50E+10	43	1	—

Tabla 43. Resultados para *test* de irradiación estático y dinámico.

Comparando dos dispositivos CRII, y teniendo en cuenta que la tecnología es la misma, se espera que un número mayor de SEFIs vaya a aparecer en el dispositivo más grande. Sin embargo, los resultados muestran que, debido al espesor diferente del encapsulado, los protones se detienen en el Si en el dispositivo más grande CRL, y se produce un menor número de SEFIs con respecto a CRS.

Con respecto a los resultados de las pruebas estáticas, el encapsulado resultó tener también un papel importante. Los SEUs se detectaron con protones a energías de 16,1 y 14,7 MeV en el dispositivo más pequeño CRS, mientras que no se detectó ningún

[5] La autora quiere agradecer a Yolanda Morilla y Mª Carmen Jiménez su aportación realizando estas pruebas con SRIM2008 y otros simuladores de materiales.

SEU en el dispositivo más grande, CRL. Con una energía de 17,3 MeV, se observa un mayor número de SEUs en el CRL que en el CRS, pero esto puede deberse a los efectos del TID.

Se puede realizar un análisis de los datos con los resultados obtenidos en las pruebas dinámicas. Un informe de cuántos SEFIs han sido detectados a lo largo de la campaña de pruebas ayudará a comprender mejor la dinámica de la degradación del dispositivo y cuando el TID está afectando al dispositivo testeado. El número total de los SEFIs detectados en los dispositivos CRII, con la energía de protones de 17,3 MeV, está representado en la Figura 45. Cuanto más grande es el dispositivo, más notorio es el efecto de TID. Para una alta tasa de dosis de energía acumulada el número de SEFIs es mayor en el dispositivo más pequeño debido a que en el encapsulado no se detienen tantos protones como en un dispositivo más grande. Sin embargo, para valores de la dosis baja, mayor número de SEFIs se observa en el dispositivo más grande, como era de esperar, ya que los efectos de TID todavía no aparecen. Para los *tests* de irradiación con protones de energía tan baja, la eliminación total o parcial del encapsulado podría ser muy conveniente para el estudio de SEEs en la tecnología probada.

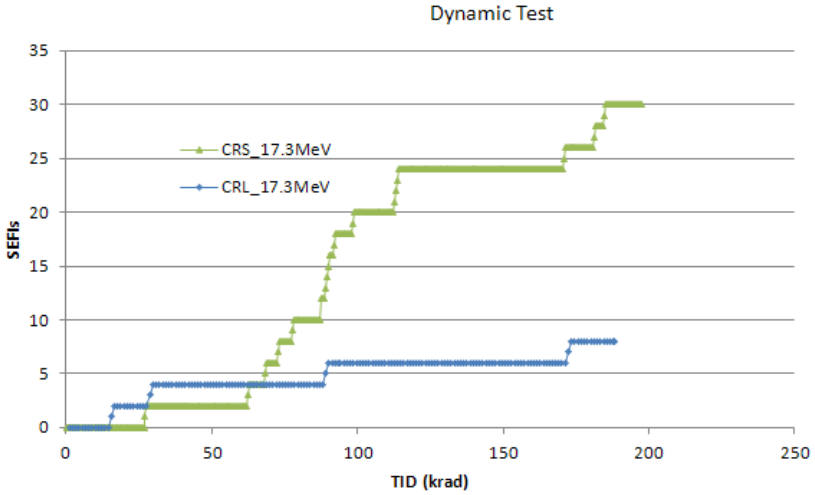


Figura 45. El número de los SEFIs acumulados frente a la dosis de energía absorbida en krad durante el *test* dinámico para dispositivos CRL y CRS de 17.3MeV.

Algunos consejos deben de tenerse en cuenta a la hora de irradiar a bajas energías. En primer lugar, el encapsulado del dispositivo absorbe una parte importante de la energía de protones. Encapsulados gruesos pueden proteger por completo los dispositivos. Por otro lado, el flujo ha de ser alto si se quiere obtener un número significativo de SEEs. Además, es aconsejable también disponer de varios dispositivos durante la prueba.

Se puede concluir que, es factible el uso de la instalación de CNA con protones de baja energía para descartar dispositivos muy sensibles y para probar los procedimientos de los *tests* de irradiación.

5.2.3. Campaña de irradiación 3

5.2.3.1. Implementación

Para probar la metodología propuesta en esta tesis doctoral se ha construido un sistema digital distribuido. El sistema se compone de cinco nodos conectados entre sí en una red, los nodos se encargan de realizar varías tareas comunes y algunas cargas útiles propias. Los nodos están conectados por un protocolo LIN, pero cualquier otro protocolo de comunicación también se podría aplicar. En la red hay cuatro nodos

esclavos y un nodo maestro. El nodo principal maestro se compone de un microcontrolador 8051, una interfaz para la conexión de algunos periféricos (GPIO, SPI, UART, etc.) mediante bus APB y un módulo controlador con el protocolo LIN implementado que se comunica con el microcontrolador por medio de APB, Figura 46. El micro 8051 se encarga de configurar la red LIN y de obtener los datos de los nodos esclavos de la red. Este micro fue seleccionado debido a su uso general en aplicaciones de automoción, industriales y aeroespaciales.

Los nodos esclavos se componen de un módulo de controlador LIN junto con el hardware específico para las operaciones de cargas útiles y para las tareas críticas globales. Una de las tareas críticas reconocida a nivel mundial y que se utiliza en muchos sistemas electrónicos distribuidos es el tiempo real. En esta tesis, un reloj de tiempo real robusto (RRTC - *Robust Real Time Clock*) se ha diseñado, en el que cada contador se duplica y se compara con el fin de realizar un seguimiento de la presencia de SEUs durante las campañas de inyección de fallos o durante el funcionamiento en campo.

En el sistema integrado, el nodo maestro, incluye un votador NMR con la detección y corrección de la réplica defectuosa, para el tiempo real. También, un RRTC está incluido en el nodo maestro. Las principales tareas para la configuración de la red, recolección y análisis de los datos del tiempo real se ejecutan en el microcontrolador 8051, mientras que cada reloj de tiempo real con su correspondiente informe de error interno y la votación de mayoría la realizan los bloques de hardware específicos.

Las campañas de inyección de fallos realizadas fueron dirigidas a la observación de los efectos de los fallos en los bloques específicos del hardware, así como al análisis de la solidez de las tareas de mantenimiento de tiempo real y a las capacidades de la comunicación. Se han identificado diferentes tipos de errores:

• SEUs en los bloques del tiempo real robusto – RRTC. Estos fallos pueden afectar a cualquier réplica de RTC. Cuando se afecta a la primera réplica (utilizada para el mantenimiento global del Tiempo Real), un informe de error incrementa el valor acumulado de errores y el votador global NMR informa de qué nodo está fallando. Una vez recibido este informe todos los nodos se corrigen con una acción de corrección global establecida. Cuando los fallos afectan a la segunda réplica, el valor acumulado de errores se incrementa, pero el votador global NMR no informa de ningún error. No se realiza ninguna acción global de corrección en este caso.

• SEUs en el bloque votador NMR. Cuando el votador NMR informa de un error, pero los informes de error de los nodos individuales se encuentran correctamente, esto significa que la ubicación de los SEUs está en los elementos de memoria situada en el nodo maestro, donde se realiza la votación por mayoría. Estos fallos hay que compararlos con el funcionamiento normal del sistema distribuido, sin fallos, ya que pueden aparecer algunos retrasos debidos al protocolo de comunicaciones pero que se corrigen con el votador NMR.

• SEUs en el módulo controlador LIN. Estos errores se deben a los SEUs en los elementos de la memoria de los módulos del controlador LIN.

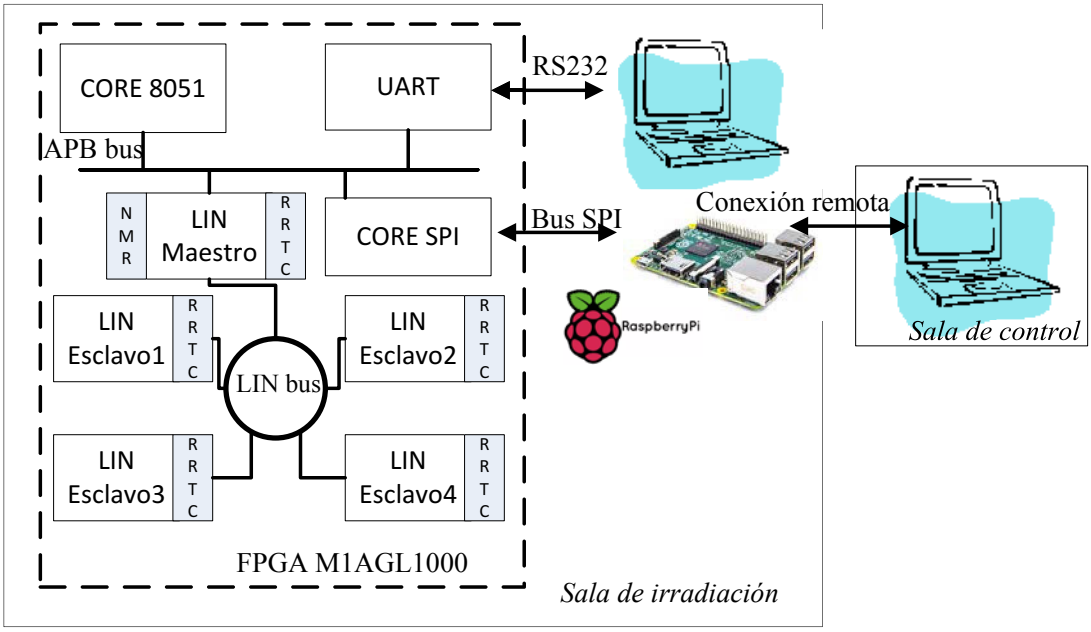


Figura 46. El sistema electrónico distribuido diseñado para campañas de inyección de fallos en instalaciones del CNA

5.2.3.2. Resultados experimentales

Dispositivo a testear

El sistema distribuido se ha prototipado en varias FPGAs Igloo, basada en la tecnología Flash, (M1AGL1000-FGG484) de Microsemi®, siendo una tecnología de 130 nm. La memoria de configuración en esta tecnología es robusta frente a la irradiación de protones, como se ha comprobado en las campañas de irradiación realizadas anteriormente en el CNA.

La energía emitida por el ciclotrón se había ajustado al valor de 18 MeV y se utilizó la línea de un haz externo. La placa con el circuito bajo prueba se situó a unos 110 mm de la boquilla de la salida del haz y se utilizó una lámina de tungsteno de 25µm de ventana, de modo que la energía final que llegaba a la superficie del circuito de prueba fue de 16.95 MeV. Este valor final de la energía del haz incidente se ha obtenido a partir de los datos de las pérdidas de energía mediante el cálculo realizado por el programa SRIM 2013 [109].

A los dispositivos a testear se les retiró el encapsulado, para poder lograr la penetración directa del haz a un área de 200 micras de silicio puro. El seguimiento del flujo de protones se realizó de manera indirecta de forma que la corriente del haz se midió en el grafito aislado eléctricamente del colimador detrás de la ventana de salida. Los valores del flujo variaban en un rango del 5% durante cada ejecución. La dosis absorbida correspondiente a cada fluencia está mostrada en la Tabla 44.

Fluencia (p/cm2)	TID (krads)
4,10E+11	156
1,70E+11	65
4,30E+11	163
6,60E+11	250
8,70E+11	330

Tabla 44. Características del material.

Para un análisis más completo de los efectos de la irradiación, a lo largo de la campaña de radiación se estudió la composición del material interno de las FPGAs utilizadas (Microsemi ™ Igloo)[6].

Un microscopio electrónico de barrido, SEM (*Scanning Electron Microscope*), se ha utilizado para el análisis de las muestras cortadas transversalmente. Se han podido identificar diferentes materiales (metales y óxidos) con sus diferentes grosores. En la Figura 47 se muestra una imagen de la sección transversal obtenida. Los contactos en forma de bola, fabricadas con oro, se pueden apreciar claramente en la parte superior de la sección y se encuentran montados en el circuito fabricado. Estos contactos con un mayor nivel de detalle se muestran en la Figura 48. Por otra parte, se pueden observar cuatro regiones diferentes en el dispositivo encapsulado con diferentes concentraciones de silicio y otros elementos, así como diferentes texturas. Las regiones A y B corresponden a materiales compuestos y al sustrato del circuito, mientras que las regiones C y D constituyen el circuito en sí.

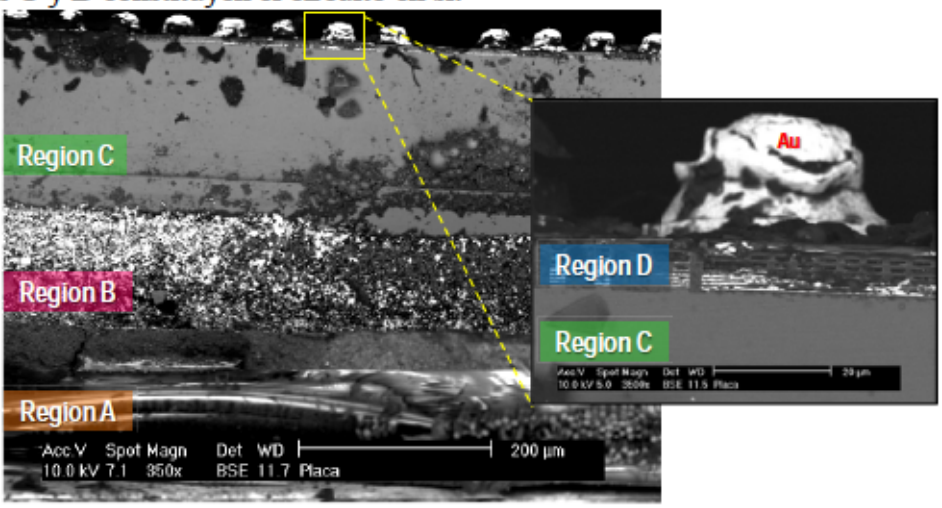


Figura 47. Sección transversal de la FPGA Igloo.

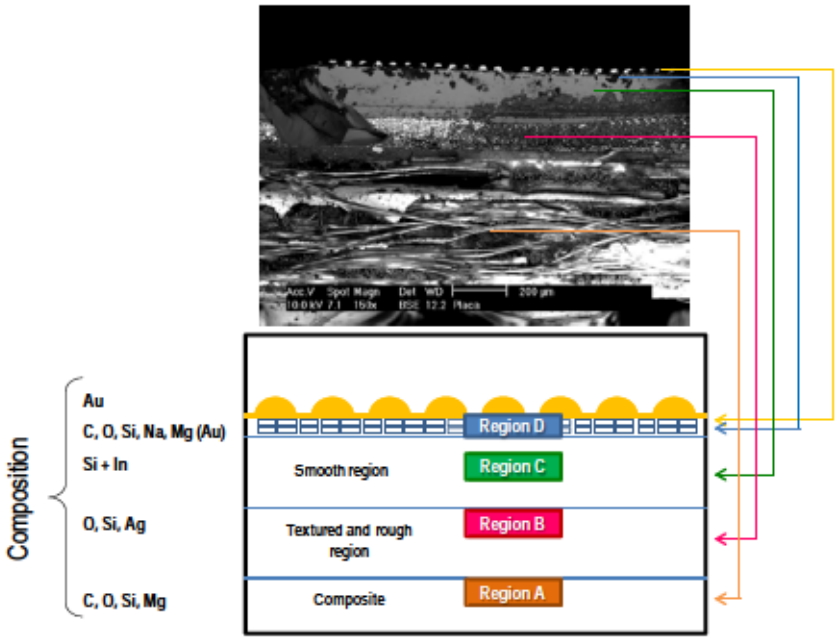


Figura 48. La composición del dispositivo bajo *test*.

[6] La autora quiere agradecer al Departmento de Ciencia e Ingeniería de los Materiales e Ingeniería Química que realizó este estudio en su microscopio electrónico de barrido (SEM)

Se ha medido el espesor de estas regiones, como su composición, con el fin de realizar varias simulaciones con el programa SRIM2013 y así poder comprobar la longitud de penetración y a continuación calcular el LET de los protones durante la campaña de inyección de fallos, Figura 49, Figura 50.

La energía del haz de protones de salida fue de 17,3 MeV, pero debido a las pérdidas, la energía en la superficie del DUT resulto ser de 16.95 MeV. A medida que el encapsulado de los dispositivos se fue removiendo, el haz penetraba directamente en la región activa a través de los 200 µm de silicio puro.

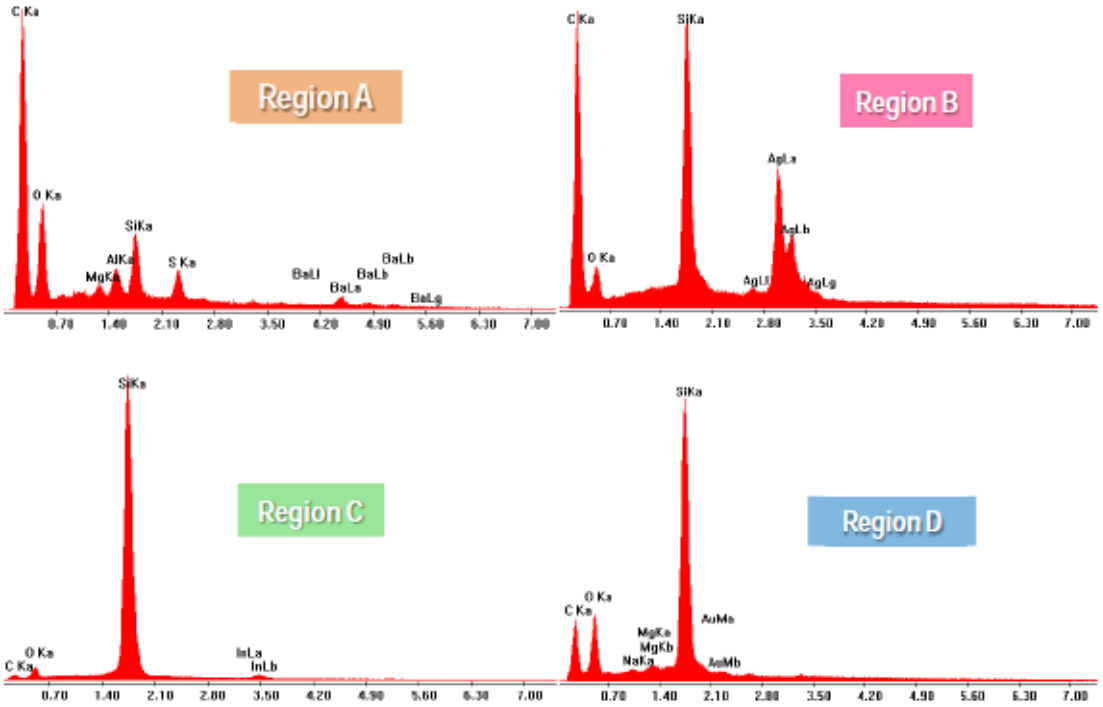


Figura 49. La composición del material en distintas regiones del circuito bajo *test*.

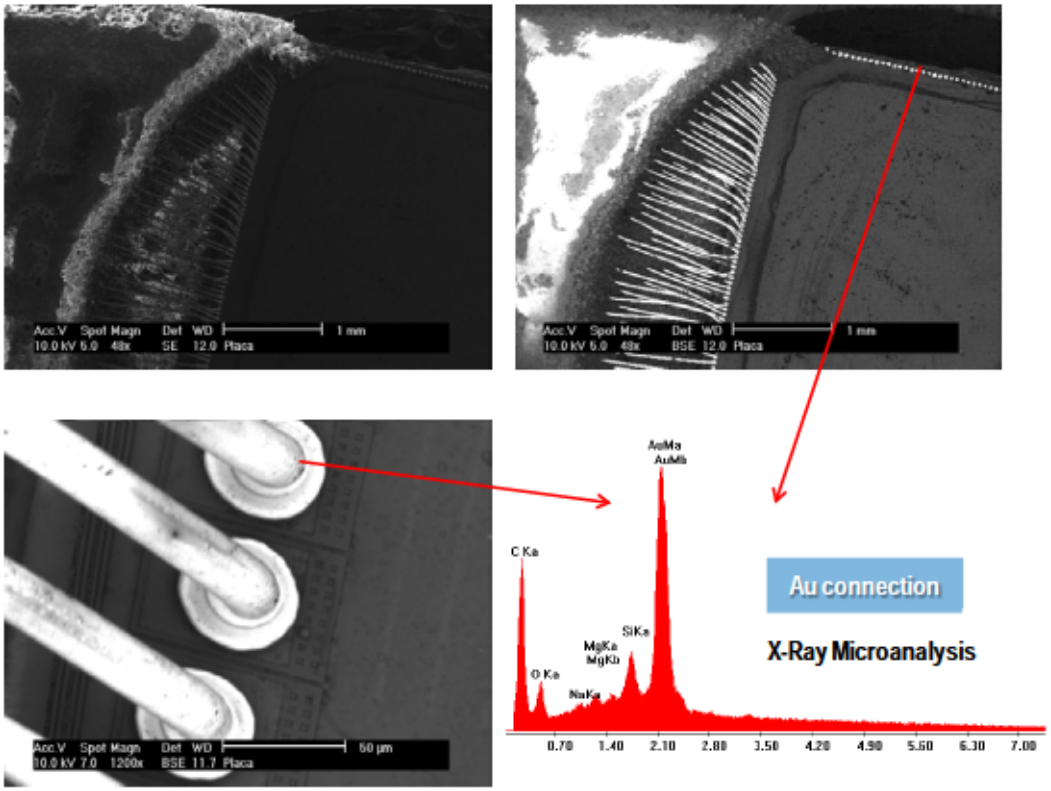


Figura 50. Composición de los conectores en el circuito bajo *test*.

Campaña de Inyección de Fallos mediante Irradiación

La campaña de inyección de fallos mediante irradiación con haz de protones se llevó al cabo sobre el sistema distribuido implementado. Los experimentos se han realizado en un acelerador de protones, ciclotrón compacto 18/9, del Centro Nacional de Aceleradores de Sevilla. El sistema ha sido sometido bajo un haz de protones con el fin de provocar los SEUs en los elementos de memoria. En la instalación se han usado energías de hasta 18 MeV siendo este el valor máximo disponible en la instalación. Previamente a los experimentos de irradiación, los encapsulados de las FPGAs probadas fueron retirados, con el fin de asegurar que la energía más alta pueda llegar al circuito integrado.

El sistema distribuido a testear está constituido por cinco nodos, uno maestro y cuatro esclavos que se separan en dos FPGAs idénticas M1AGL1000 los que están interconectados entre sí por el bus LIN, Figura 53. Se ha separado los nodos para realizar un análisis más preciso de las causas de los fallos del sistema. La FPGA con el nodo maestro tiene también un microprocesador 8051 el que ejecuta el programa *software* del *test* de sistema. El programa del microprocesador incluye el envío del tiempo real (TR) de todos los esclavos por la UART y del informe final con el tiempo real votado y corregido por el SPI, Figura 51.

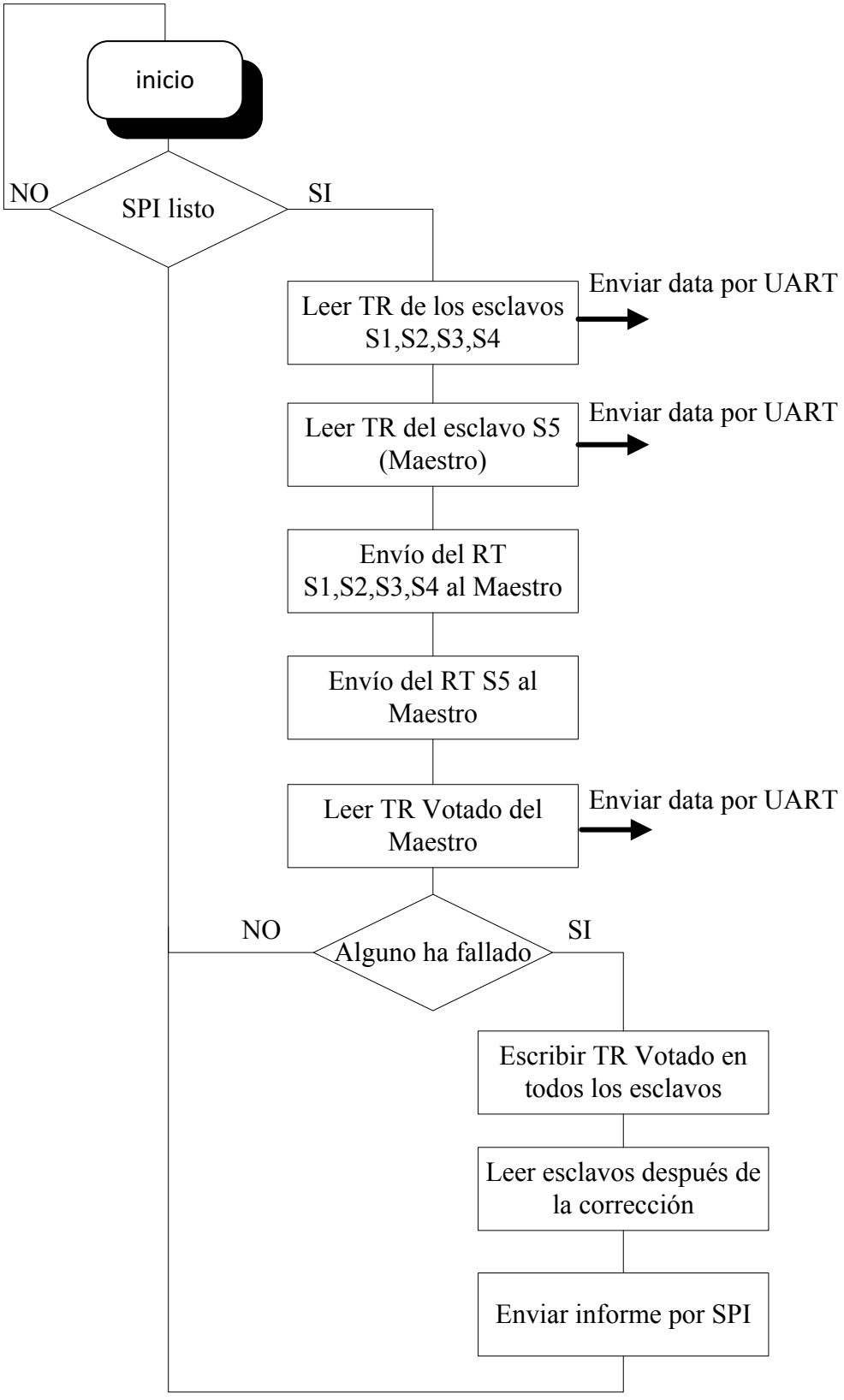


Figura 51. Diagrama de flujo del 8051 con el programa del *test* del sistema.

Durante la irradiación del circuito con haz de protones fue realizada una verificación continua de la actividad del sistema, por medio de una conexión remota con un ordenador en la sala de control. Por medio del módulo SPI, incluido como periférico del microprocesador del nodo maestro, el programa se conecta con otro microprocesador ARM situado en una placa Raspberry Pi, con el fin de permitir la

comunicación entre el sistema distribuido bajo *test* y la sala de control. Así mismo, por el puerto UART se enviaban datos en tiempo real de los esclavos y de los nodos con fallo. En los resultados finales se comprobaron y compararon los dos informes obtenidos durante la campaña.

La FPGA con cuatro nodos esclavos, sólo tiene cuatro controladores de bus LIN interconectadas entre sí (endurecidos y con la tarea distribuida del tiempo real). Todos los esclavos del sistema tienen un hardware que lleva la cuenta de los milisegundos y segundos del tiempo real del nodo.

Prueba dinámica de los esclavos

En primer lugar, para las pruebas dinámicas de los nodos esclavos se ha elegido la FPGA Igloo pequeña (IGS-1), empezando a irradiar desde una energía de 17,3 MeV y bajándola. El circuito ocupaba alrededor de 20% de área disponible de la FPGA.

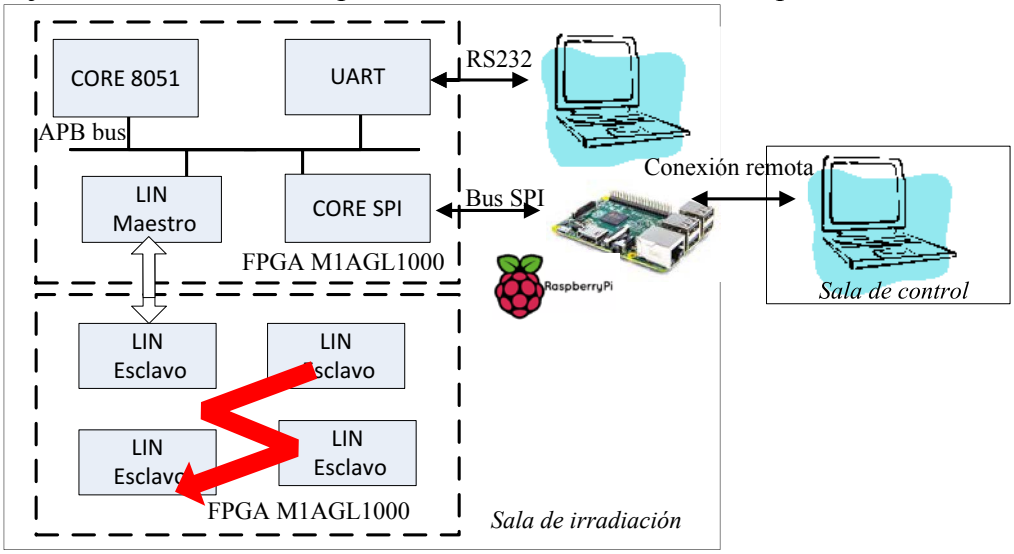


Figura 52. Esquema del sistema distribuido a radiar, nodos esclavos

Al principio se ha configurado el ciclotrón de tal forma que la muestra recibía una fluencia de $1 \cdot 10^8$ p / cm^2 durante 9.709 segundos y después se subió el flujo al $1 \cdot 10^9$ p / (s · cm^2) durante 808 s. LET $2{,}28 \cdot 10^{-2}$(MeV/mg/cm^2),

Test DUT1	***Tiempo (seg)***	***SEU/ SEFIs***	***Acción***
1	2.981	Falla esclavo 3 (ES3)	Reset, stop
2	4.206	Falla ES3 y la comunicación (EC)	Reset, stop
3	2.522	ES3	Stop
4	73	2 fallos del ES1	
4	121	25 fallos del ES1	Reset
4	103	EC	Reset, stop
5	9	ES4	Reset
5	47	EC	Reset, stop
6	120	ES3	
6	133	EC	Reset, stop
7	1	EC	Reset, stop
8	1	EC	Reset, stop

Tabla 45. Los resultados del *test* dinámico de los esclavos.

Después de irradiar el sistema alrededor de 120 minutos con un flujo en torno al valor de $1 \cdot 10^8$, se han detectado fallos del mantenimiento del tiempo real en el esclavo 3, que no se corregían. También se ha detectado 1 SEU (SEFI) debido a la comunicación en la red. Tras unos 10 minutos con el flujo en torno al valor de $1 \cdot 10^9$ aparecen 6 SEUs en los esclavos 1, 3 y 4 en sus bloques del tiempo real RRTC. Además, falló el tiempo real enviado, que se corrigió. El indicador del esclavo fallido también se activó una vez. Además, se detectaron 5 SEFIs en la comunicación del protocolo LIN. La mayoría de los SEUs llegaron a detectarse con una fluencia más alta.

Prueba dinámica 1 del nodo Maestro

Tras irradiar la FPGA con los nodos esclavos, se sustituyó la FPGA por otra de la misma tecnología, pero con el diseño del controlador maestro Figura 53, la muestra IGM-3, y ésta fue irradiada también con la energía de 17,3 MeV. El encapsulado removido de esta FPGA era de plástico. El sistema con el nodo maestro tenía una ocupación del 60% del área disponible de la FPGA.

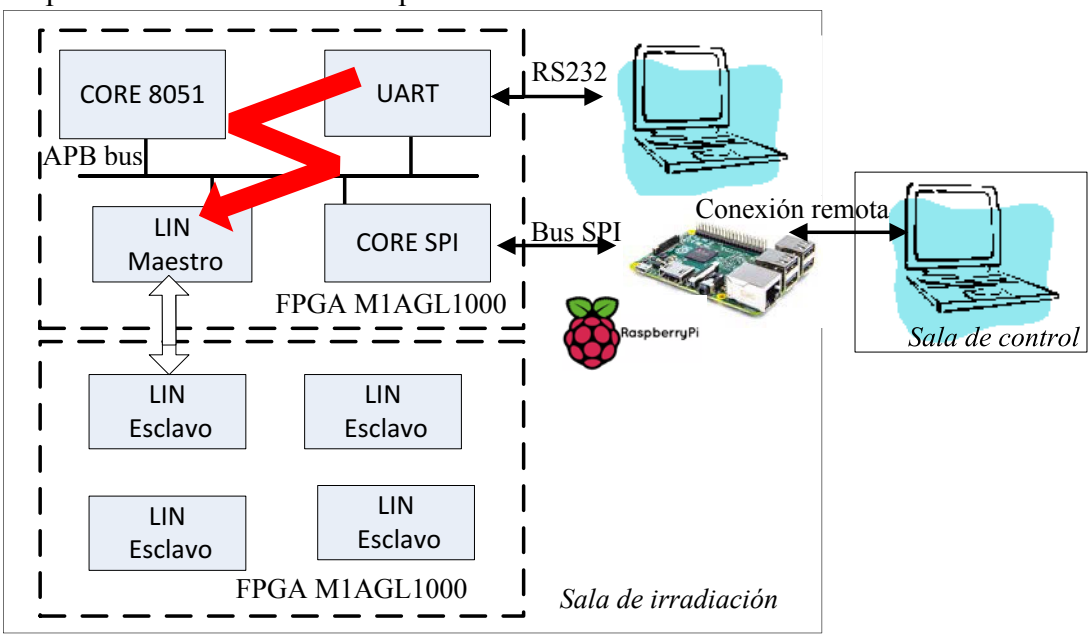


Figura 53. El esquema del sistema a radiar, el nodo maestro.

El flujo recibido en esta prueba fue de $5 \cdot 10^8$ p / (s·cm^2) durante un tiempo de 4.955s. El bloque de tiempo real robusto implementado en el nodo maestro ha informado acerca de 4 SEUs y se detectaron 3 errores SEFI en la comunicación del protocolo LIN.

Test DUT2	*Time (seg)*	*SEU/ SEFIs*	*Acción*
1	359	EM	
1	705	EC	Reset, stop
2	361	EM	Corrección
2	742	EM	Corrección
2	1.124	EM	Corrección
2	1.663	2 fallos en ES3	Reset, stop
3	1	EC	Reset, stop
4	1	EC	Reset, stop

Tabla 46. Los resultados del *test* dinámico del maestro 1.

Con un flujo cuyo valor estaba en torno a $5 \cdot 10^8$ p / (s · cm^2), durante 40 minutos, se han detectado 4 posibles SEUs que provocaron fallos en el mantenimiento del tiempo real, bien debido a que la votación en el maestro ha sido errónea o bien porque no se interpretó correctamente qué nodo estaba fallando. Otros 2 SEUs (SEFIs) se han detectado en la comunicación de la red LIN y el maestro no ha corregido los tiempos reales de los esclavos. En este momento ha sido necesario reiniciar el sistema para poder seguir funcionando. Pero después del reinicio se verificó que el daño había sido importante, habiéndose roto la FPGA sin la posibilidad de poder arrancar de nuevo el sistema.

Prueba dinámica 2 del nodo Maestro

Una segunda campaña de radiación con el mismo circuito de nodo maestro se ha implementado en otra FPGA de la misma tecnología, correspondiente a la muestra IGM-2. La muestra recibió un flujo de $2 \cdot 10^8$ p / (s · cm^2) durante 13.788 segundos. El bloque de tiempo real robusto implementado dentro del nodo maestro ha informado sobre 9 SEUs y se detectó 1 error SEFI en la comunicación LIN.

Test DUT3	Time (seg)	SEU/ SEFIs	Acción
1	160	EM	
1	541	EM	
1	727	EM	
1	922	EM	
1	1.304	EM	
1	1.456	EM	
1	1.685	EM	
1	2.067	EM	
1	2.448	EM	
1	2.478	EC	Reset, stop
2	296	EM	
2	530	EM	
2	677	EM	
2	1.058	EM	
2	1.440	EM	
2	1.821	EM	
2	2.165	EC	Reset, stop
3	372	EM	
3	718	EC	Reset, stop
4		EC	Reset, stop

Tabla 47. Los resultados del *test* dinámico del maestro2

Tras la irradiación durante un tiempo de unos 93 minutos con un flujo con un valor en torno a $2 \cdot 10^8$ p / (s · cm^2), 16 posibles SEUs del mantenimiento del tiempo real han sido detectados. También se han detectado 3 SEUs (SEFIs) de comunicación en la red LIN con los nodos esclavos. Se ha procedido al reinicio del sistema y se pudo continuar irradiando el circuito ya que el sistema se recuperó correctamente tras reiniciarse.

5.2.3.3. Resultados

Se han irradiado 3 dispositivos Igloo® M1AGL1000. El primer dispositivo DUT1 se ha configurado con cuatro esclavos, mientras que otros dos DUT2 y DUT3 se ha configurado solo con el nodo maestro. Se ha realizado un funcionamiento continuo de intercambio de los datos en el sistema distribuido mientras los dispositivos estaban bajo irradiación. Un resumen de los resultados obtenidos para los esclavos y para el maestro de forma separada se muestra en la Tabla 48 y la Tabla 49.

Test DUT1	Flujo ($p/cm^2 \cdot s$)	Fluencia (p/cm^2)	TID (krads (Si))	Errores
Run1	$5{,}7 \cdot 10^7$	$4{,}1 \cdot 10^{11}$	156	El informe del error en el esclavo, RRTC correcto → retardos en el bus (no hay SEU)
Run2	$6{,}6 \cdot 10^7$	$1{,}7 \cdot 10^{11}$	65	No hay errores
Run3	$7{,}3 \cdot 10^8$	$4{,}3 \cdot 10^{11}$	163	El informe de 7 errores del RRTC #esclavos, #instantes → SEUs 5 SEFIs en el protocolo LIN

Tabla 48. Resultados del *test* de radiación para los esclavos

DUT	Flujo ($p/cm^2 \cdot s$)	Fluencia (p/cm^2)	TID (krads (Si))	Errores
DUT2	$2{,}7 \cdot 10^8$	$6{,}6 \cdot 10^{11}$	250	El informe de 4 errores del RRTC en el nodo maestro → SEUs 3 SEFIs en el bus de comunicación LIN→ SEUs
DUT3	$1{,}6 \cdot 10^8$	$8{,}7 \cdot 10^{11}$	330	El informe de 9 errores del RRTC en el nodo maestro → SEUs 1 SEFI en el bus de comunicación LIN→ SEU

Tabla 49. Resultados del *test* de radiación para el maestro.

Como se puede apreciar, en las tablas de resultados, el número de fallos observados fue pequeño. Este hecho es debido a que la tecnología *Flash* es muy poco sensible a los efectos de la radiación y, por tanto, se puede decir que los fallos sólo afectaban a los flip-flops internos. Esto nos permite evaluar adecuadamente las técnicas de mitigación utilizadas. Además, los resultados obtenidos en la campaña de inyección de fallos mostraron que el endurecimiento colaborativo [111] está corrigiendo adecuadamente el sistema y elimina los errores debidos a los SEUs (así como al ruido o retrasos en el enlace de comunicación) en los elementos de memoria del sistema distribuido, por lo que resulta ser válido para las aplicaciones críticas.

Capítulo 6 CONCLUSIONES

El endurecimiento de sistemas digitales con distribución de la funcionalidad se hace cada vez más necesario debido al aumento de aplicaciones de este tipo en los entornos espaciales, de aviónica y de la automoción, en los cuales los efectos de la radiación ionizante son notorios. La disminución del tamaño de los transistores hace a los circuitos y sistemas más susceptibles ante la radiación. Por otro lado, los sistemas digitales son cada vez más complejos lo que a su vez requiere de utilización de unas metodologías nuevas para poder realizar un óptimo endurecimiento al nivel de sistema. En esta tesis doctoral se propone una nueva metodología para el efectivo endurecimiento de un sistema digital distribuido. La metodología propuesta comprende el estudio del sistema para determinar sus partes críticas mediante técnicas de inyección de fallos con una Emulación Autónoma, el endurecimiento del sistema y la posterior irradiación del sistema endurecido para testear y verificar su funcionamiento antes de someterlo a un entorno crítico. En la inyección de fallos mediante emulación *hardware* se ha ampliado la herramienta de Emulación Autónoma, desarrollada en el grupo de investigación DMA-UC3M, para permitir la inyección de fallos múltiples en elementos de memoria y para permitir el análisis detallado del efecto acumulado de los fallos detectados mediante técnicas de mitigación de fallos. Para el endurecimiento del sistema se han propuesto varias técnicas locales y globales, que permiten realizar un seguimiento de la aparición de los fallos, localización de su origen, estudio de su efecto acumulado y comprobación de su eliminación y/o detección mediante técnicas de mitigación de fallos. Entre estas técnicas se destaca el bloque votador por mayoría con localización de la réplica fallada y el detector de minoría, que permiten identificar los nodos de la red distribuida que están fallando. Finalmente, en la cualificación del sistema mediante campañas de irradiación con aceleradores de partículas (u otro método equivalente validado por las agencias espaciales) se ha propuesto un método de planificación y realización de la campaña que permite la monitorización continua del efecto de los fallos. Este método se puede extrapolar a la realización de *test on-line* durante el funcionamiento normal del sistema distribuido, para la detección de fallos procedentes de la radiación ionizante (fallos transitorios) y fallos debidos al envejecimiento de los dispositivos (fallos permanentes).

Durante el estudio y el desarrollo de la citada metodología se han analizado diferentes dispositivos FPGAs/CPLDs evaluándose su uso y fiabilidad en los entornos críticos. Dicho análisis ha permitido detectar los puntos fuertes y débiles de circuitos digitales prototipados en los citados dispositivos, proponiendo las técnicas de endurecimiento adecuadas para los mismos, así como caracterizar la robustez de la tecnología en sí. Se han implementado distintos circuitos y sistemas para efectuar un análisis adecuado tanto de las técnicas de endurecimiento como de los dispositivos a testear. Durante el estudio se ha aumentado de forma gradual la complejidad del sistema a endurecer, partiendo de los componentes más básicos y llegando a un sistema distribuido complejo.

En este trabajo se ha tomado como ejemplo del sistema distribuido a endurecer un sistema de comunicación serie basado en el protocolo LIN que además se ha

comparado con otro protocolo de comunicación CAN. Los protocolos LIN y CAN se complementan entre sí y permiten unir todos los dispositivos electrónicos de un vehículo en una única red multifuncional de a bordo. Además, el bus CAN se utiliza en las áreas que requieren una gran fiabilidad y alta velocidad. En cambio, el uso del bus LIN resulta adecuado para la comunicación entre unidades electrónicas menos costosas, que trabajan con velocidades medias/bajas de transmisión de datos a unas distancias cortas que al mismo tiempo proporcionan una flexibilidad, versatilidad y facilidad del desarrollo y la depuración.

Se ha desarrollado un sistema distribuido básico con sus actividades fundamentales de intercambio y análisis de los datos. El sistema implementado contiene un nodo maestro y múltiples nodos esclavos. El microprocesador por medio de software da órdenes a la red distribuida y solo se comunica con el nodo maestro. La comunicación entre los nodos se realiza mediante el protocolo LIN, que a su vez tiene una interfaz con el bus APB para recibir/enviar la información al microprocesador. El sistema es totalmente síncrono, configurable y versátil. Los nodos esclavos tienen las mismas tareas para poder verificar el correcto funcionamiento del sistema en condiciones de radiación ionizante. Se ha implementado un circuito *Robust Real Time Clock* (RRTC), el que se duplica cada contador y se compara con el fin de realizar un seguimiento de la presencia de SEUs durante las campañas de inyección de fallos.

En este sistema desarrollado se han incluido las técnicas de monitorización y rastreo de la presencia, el efecto y la mitigación de los fallos. El nodo maestro lleva un votador NMR que detecta y corrige la réplica defectuosa y mantiene el correcto funcionamiento del RRTC de los esclavos y el suyo.

Se ha realizado la verificación de la sensibilidad a la radiación de los componentes mediante la emulación. Una vez localizados los bloques más débiles mediante la emulación, se realiza el endurecimiento y el posterior ensayo de irradiación para la medida de la sensibilidad a la radiación de los sistemas distribuidos. Se ha endurecido el sistema teniendo en cuenta los requisitos del coste y prestaciones, aplicando las técnicas de endurecimiento existentes y el endurecimiento colaborativo, que permite utilizar componentes menos robustos en la red distribuida pero que replican las tareas críticas y disminuyen de forma destacada la sensibilidad. Durante la puesta a punto del sistema a irradiar, se identificó como crítica la comparación entre la robustez de los dispositivos con distinta tecnología, encapsulado, capacidad del dispositivo y el número de pines. Como resultado de estas consideraciones, se elaboró una técnica, basada en un bloque de detección de minoría, que permitía la comparación de los datos obtenidos tras la irradiación de una forma directa y transparente al sistema de *test*. Para verificar de forma experimental los resultados, se realizó una segunda campaña de irradiación en el CNA. Esta investigación con el título *"SEU Sensitivity Comparison for Different Reprogrammable Technologies with Minority Check Block"* y sus resultados se publicaron en la revista *Transactions on Nuclear Science*.

Se ha implementado el sistema en las placas de la familia Microsemi® y Xilinx®. Se ha desarrollado un procedimiento de irradiación, probando distintos dispositivos durante varias compañas en el centro de CNA. Los dispositivos han sido elegidos después de un análisis de varias tecnologías que utilizan en los entornos críticos. La tesis doctoral ha sido completada con una compaña de irradiación del sistema distribuido desarrollado construido con los dispositivos programables comerciales FPGAs Igloo de Microsemi®, elegido gracias a que es una tecnología más robusta basada en la tecnología Flash.

Referencias

[1]. Coulouris, G. F., and Dollimore, J., Distributed Systems: Concepts and Design, Addison-Wesley Publishing, 1988

[2]. M. Santarini "Cosmic Radiation Comes to ASIC and SOC Design" EDN Europe, May 2005 (www.edn.com).

[3]. Coderre, Jeffrey. 22.01 Introduction to Ionizing Radiation. Fall 2006. (Massachusetts Institute of Technology: MIT OpenCourseWare), http://ocw.mit.edu (Accessed 27 March, 2014). License: Creative Commons BY-NC-SA

[4]. L. Pouponnot "Strategic Use of SEE Mitigation Techniques for the Development of the ESA Microprocessors: Past, Present, and Future" 11th IEEE International On-Line Testing Symposium, 2005.

[5]. R. D. Schrimpf, "Radiation Effects in Microelectronics". En R. Velazco, P. Fouillat, R. Reis (Eds.) "Radiation Effects on Embedded Systems", Holanda, Springer, 2007, pp. 11-29.

[6]. International Technology Roadmap for Semiconductors (ITRS). Ediciones 2001-2015.

[7]. Jean-Marie Lauenstein Code 561, NASA Goddard Space Flight Center, Test Standard Revision Update: JESD57, "Procedures for the Measurement of Single-Event Effects in Semiconductor Devices from Heavy-Ion Irradiation" NEPP Electronic Technology Workshop June 23-26, 2015 NASA Goddard Space Flight Center in Greenbelt, MD

[8]. M. Portela, Tesis Doctoral "Técnicas de inyección de fallos basadas en FPGAs para la evaluación de la tolerancia a fallos de tipo SEU en circuitos digitales", Madrid, 2007.

[9]. L. Pouponnot "Strategic Use of SEE Mitigation Techniques for the Development of the ESA Microprocessors: Past, Present, and Future" 11th IEEE International On-Line Testing Symposium, 2005.

[10]. R. C. Baumann, "Radiation-Induced Soft Errors in Advances Semiconductor Technologies", IEEE Transactions on Device and Materials Reliability, Vol. 5, No. 3, pp. 305-316, Septiembre 2005.

[11]. www.jedec.org

[12]. www.cnes.fr

[13]. Klutke, G.; Kiessler, P.C.; Wortman, M.A. "A critical look at the bathtub curve". IEEE Transactions on Reliability 52 (1): 125–

129. doi:10.1109/TR.2002.804492. ISSN 0018-9529. Retrieved 2015-06-16.

[14]. A.H. Johnston, S.M. Guertin, "The effects of space radiation on linear integrated circuits", Aerospace Conference Proceedings, 2000 IEEE, Volume 5, pp. 363-369, March 2000

[15]. G C. Messenger and M. S. Ash, "Single Event Phenomena", International Thompson Publishing, U.S.A, 1997.

[16]. R. Baumann, "Nuclear and Space Radation Effects Conference Short Course Notebook", Insitute of Electrical Electronics Engineer, Cap. II, July 2005.

[17]. I. López Calle, Tesis doctoral "Emulación de los efectos de la radiación ionizante en dispositivos analógicos mediante láser pulsado de femtosegundo sintonizable." Madrid 2010

[18]. S. Kuboyama, T. Suzuki, T. Hirao and S. Matsuda, "Mechanism for SingleEvent Burnout of Bipolar Transistor", IEEE Transactions on Nuclear Science, Vol. 47, nº. 6, pp. 2634-2639, December 2000.

[19]. C. Barillot, O. Serres, R. Marec and P. Calvel, "Effects of reliability screening tests on bipolar integrated circuits during total dose irradiation", IEEE Transactions on Nuclear Science, Vol. 45, nº 6, pp. 2638-2643, December 1998.

[20]. R. Jones, A. M. Chugg, C. M. S. Jones, P. H. Duncan, C. S. Dyer, and C. Sanderson, "Comparison between SRAM SEE cross-sections from ion beam testing with those obtained using a new picosecond pulsed laser facility," IEEE Transactions on Nuclear Science, vol. 47, pp. 539–544, June 2000.

[21]. M. Nicolaidis "Design for Soft Error Mitigation" IEEE Transactions on Device and Materials Reliability"

[22]. Johnson Johnson B.W. y Addison Wesley "Design and analysis of fault tolerant digital systems", 1989.

[23]. D.K. Pradhan "Fault-Tolerant Computer System Design" Prentice Hall, 1996.

[24]. Pratt Brian Improving FPGA Design Robustness with Partial TMR: Reliability Physics Symposium Proceedings: IEEE International, 2006.

[25]. M. Berg "Complexity Management and Design Optimization Regarding a Variety of Triple Modular Redundancy Schemes through Automation" Marlug, 2010.

[26]. M. Nicolaidis "Design for Soft Error Mitigation" IEEE Transactions on Device and Materials Reliability, Vol. 5, No. 3, septiembre 2005.

[27]. P. Cheynet, B. Nicolescu, R. Velazco, M. Rebaudengo, M. Sonza Reorda, 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, diciembre 2000.

[28]. N. Oh, P. P. Shirvani, E. J. McCluskey, "Control-Flow Checking by Software Signatures", IEEE Transactions on Reliability, Vol. 51, No. 2, pp. 111-122, Marzo 2002.

[29]. R. Koga, P. Yu, K. B. Crawford, S. H. Crain and V. T. Tran, "Permanent single event functional interrupts (SEFIs) in 128- and 256-megabit synchronous dynamic random access memories (SDRAMs),"Radiation Effects Data Workshop, 2001 IEEE, Vancouver, BC, 2001, pp. 6-13.

[30]. T. Riesgo, PhD thesis, "Modelado de fallos y estimación de los procesos de validación funcional de circuitos digitales descritos en VHDL sintetizable", Universidad Politecnica de Madrid, 1996.

[31]. Arlat, J.; Boué, J.; Crouzet, Y.; Jenn, E.; Aidemark, J.; Folkesson, P.; Karlsson, J.; Ohlsson, J. & Rimén, M.; Benso, A. & Prinetto, P. (ed.); Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation; 4.2: MEFISTO: A Series of Prototype Tools for Fault Injection into VHDL Models; Kluwer; 2003, pp. 177-193.

[32]. Gil, D.; Baraza, J.C.; Garcia, J. & Gil, P.J.; Benso, A. & Prinetto, P. (ed.); Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation; 4.1 VHDL Simulation-based Fault Injection Techniques; Kluwer; 2003, pp. 159-176.

[33]. H. R. Zarandi, S. G. Miremadi, A. Ejlali "Fault Injection into Verilog Models for Dependability Evaluation of Digital Systems" 2nd International Symposium on Parallel and Distributed Computing, 2003.

[34]. V, Sieh, O. Tschäche, F. Balbach. "VERIFY: Evaluation of Reliability Using VHDL-Models with Embedded Fault Descriptions", 27th International Symposium Fault Tolerant Computing, 1997.

[35]. E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, "Fault Injection into VHDL Models: the MEFISTO Tool", Proc. FTCS-24, 1994, pp. 66-75

[36]. T.A. Delong, B.W. Johnson, J.A. Profeta III, "A Fault Injection Technique for VHDL Behavioral-Level Models", IEEE Design & Test of Computers, Winter 1996, pp. 24-33.

[37]. D. Gil, R. Martinez, J. V. Busquets, J. C. Baraza, P. J. Gil, "Fault Injection into VHDL Models: Experimental Validation of a Fault Tolerant Microcomputer System", Dependable Computing EDCC-3, September 1999, pp. 191-208.

[38]. J. Boué, P. Pétillon, Y. Crouzet, "MEFISTO-L: A VHDL-Based Fault Injection Tool for the Experimental Assessment of Fault Tolerance", Proc. FTCS´98, 1998.

[39]. R. Velazco, B. Martinet, G. Auvert "Laser Injection of Spot Defects on Integrated Circuits" 1st Asian Test Symposium, pp. 158-163, Noviembre 1992.

[40]. J. Karlsson, P. Lidén, P. Dahigren "Using Heavy-Ion Radiation to Validate Fault-Handling Mechanisms" IEEE Micro, Vol. 14, Issue 1, pp. 8-23, Febrero 1994.

[41]. J. E. Taylor "Discerning the Operational State of a Vehicle's Distributed Electronic Systems from Vehicle Network Traffic for Use as a Fault Detection and Diagnosis Tool" Int. Journal of Automotive Technology, Vol. 15, No. 3, 2014.

[42]. T. Arlsan et al, "ESPACENET: A Framework of Evolvable and Reconfigurable Sensor Networks for Aerospace–Based Monitoring and Diagnostics", 1st NASA/ESA Conference on Adaptive Hardware and Systems, 2006.

[43]. P. Bernardi, L. M. Bolzani, M. Rebaudengo, M. Sonza Reorda, F. Vargas, M. Violante "A New Hybrid Fault Detection Technique for Systems-on-a-Chip" IEEE Transactions on Computers, Vol. 55, No.2, pp. 185-198, Febrero 2006.

[44]. F. Lima Kastensmidt, L. Carro, R. Reis "Fault-Tolerance Techniques for SRAM-based FPGAs" Springer, 2006.

[45]. L. Sterpone, M. Violante "A new reliability-oriented place and route algorithm for SRAM-based FPGAs" IEEE Transactions on Computers, Vol. 55, No. 6, pp. 732-744, Junio, 2006.

[46]. P. Kenterlis, N. Kranitis, A. Paschalis, D. Gizopoulus, M. Psarakis "A Low-Cost SEU Fault Emulation Platform for SRAM-Based FPGAs" 12th IEEE International On-Line Testing Symposium, pp.235-241, Julio 2006.

[47]. P. Bernardi, M. Sonza Reorda, L. Sterpone, M. Violante "On the evaluation of SEU sensitiveness in SRAM-based FPGAs" 10th IEEE International On-Line Testing Symposium, Julio 2004.

[48]. J. F. Ziegler, " Terrestrial cosmic rays," IBM J. Res. Development, vol. 40, no. 1, pp. 19–39, Jan. 1996.

[49]. A. L. Pouponnot "Strategic Use of SEE Mitigation Techniques for the Development of the ESA Microprocessors: Past, Present, and Future" 11th IEEE International On-Line Testing Symposium, 2005.

[50]. C. Ferrell, "Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators", AITR-1443, Artificial Intelligence Laboratory, MIT, Mayo 1993.

[51]. A. Vaskova, C. López-Ongil, M. García-Valderas, M. Portela-García and L. Entrena, "Evaluation techniques for on-line testing of robust systems based on critical tasks distribution," 2011 IEEE 17th International On-Line Testing Symposium, Athens, 2011, pp. 258-263.

[52]. A. Vaskova et al., "Study on the effect of multiple errors in robust systems based on critical task distribution," Radiation and Its Effects on Components and Systems (RADECS), 2011 12th European Conference on, Sevilla, 2011, pp. 463-466.

[53]. FlexRay Communications Sytems-Protocol Specification Version 2.1. FlexRay Consortium, 2003. www.flexray.com

[54]. SpaceWire standard, European Space Agency Web Page, www.spacewire.esa.int

[55]. K. Kyamakya, K. Jobmann, M. Meincke "Security and Survivability of Distributed Systems: An overview" Proceedings of IEEE Milcom, Los Angeles CA. Mayo 2000. Pp 449-454.

[56]. E. A. Strunk, J.C. Knight, M.Anthony Aiello. "Distributed Reconfigurable Avionics Architectures" 23rd Digital Avionics Systems Conference, Salt Lake City. Octubre 2004.

[57]. López, J.; Royo, P.; Barrado, C.; Pastor, E. "Modular Avionics for Seamless Reconfigurable UAS Missions" 27th Digital Avionics Systems Conference: Integrated modular avionics is the modern approach. [NJ: IEEE], 2008.

[58]. A. Shashavar, "Defining a Modular, High Speed and Robust Avionic Architecture for UAV's". Masther's Thesis. Lulea University of Technology. MSc Programmes in Engineering. Space Engineering. Dpt. of Space Science, Kiruna. August 2008.

[59]. L. Alkalai "A Roadmap for Space Microelectronics Technology into the New Millenium" Proceedings of the 35th Space Congress, Florida, USA. Feb. 1998.

[60]. B.R. Blaes "Micro-Avionics Node for Distributed Avionics System" Forum Innovative Approaches to Outer Planetary Exploration. Texas, USA, 2001.

[61]. A. Davare et al. "Period Optimization for Hard Real-time Distributed Automotive Systems" Design Automation Conference, DAC'07. California, USA, 2007.

[62]. J. Novák. "Testing of CAN Based Automotive Distributed Systems Using a Flexible of IP Functions" Feb, 2010. Internet FAQ Archives. Online Education.

[63]. M. Horauer et al. "An FPGA based SoC Design for Testing Embedded Automotive Communication Systems employing the FlexRay Protocol" Austrochip 2004, pp. 119-123, Villach - Austria, October 2004.

[64]. E. Armengaud et al. "Towards a Systematic Test for Embedded Automotive Communication Systems" IEEE T. on Industrial Informatics. Vol4. No. 3. August 2008.

[65]. I. Lora et al. "INTA PicoSatellite OPTOS: Mission, Subsystems, and Payload". 4th Annual CubeSat Developers' Workshop. 2007.

[66]. http://www.cvel.clemson.edu/auto/systems/auto-systems.html

[67]. M. Portela-García et al., "Sensitivity Evaluation Method for Aeroespace Digital Systems with Collaborative Hardening" IEEE T. on Nuclear Science. IEEE Early Access:10.1109/TNS.2011.2109397.2011.

[68]. C. López-Ongil, M. García-Valderas, M. Portela-García, L. Entrena, "Autonomous Fault Emulation: A New FPGA-based Acceleration Sytem for Hardeness Evaluation". IEEE T. on Nuclear Science. Vol54. Issue 1, part 2. Pp. 252-262, Feb. 2007.

[69]. C. López-Ongil, M. Portela-García, M. García-Valderas, A. Vaskova, L. Entrena, J. Rivas-Abalo, A. Martin-Ortega, J. Martinez-Oter, S.

Rodriguez-Bustabad, I. Arruego "SEU sensitivity of robust communication protocols". 188-193, IOLTS 2012.

[70]. W.K.Stuckey et al. "Solar Ultaviolet and Space Radiation Effects on Iflatable Materials" Space and Missile Sytems Center (8565-9), 2000.

[71]. J. López, P. Royo, C. Barrado, E Pastor, "Modular Avionics for Seamless Reconfigurable UAS Missions" 27th Digital Avionics Systems Conference: Integrated modular avionics is the modern approach. [NJ: IEEE], 2008.

[72]. R. Trautner, P. Armbruster, P. Fabry, A. Fernandez-Leon, J. Ilstad, D. Jameux, M. Suess, R. Wigand, "Avionics Architectures and Components for Planetary Entry Probe Payloads and Systems", 6th International Planetary Probe Workshop (IPPW6), Altanta, 2008.

[73]. K. Kyamakya, K. Jobmann, M. Meincke "Security and Survivability of Distributed Systems: An overview" Proceedings of IEEE Milcom, Los Angeles CA. Pp 449-454. Mayo 2000.

[74]. I. Lora et al. "INTA PicoSatellite OPTOS: Mission, Subsystems, and Payload". 4th Annual CubeSat Developers' Workshop. 2007.

[75]. LIN Consortium, "LIN Specification Package. Revision 2. 0" Sept, 2003

[76]. XAPP432 (v1.1) Xilinx Application Note "Implementing a LIN Controller on a CoolRunner-II CPLD", April 3, 2007.

[77]. Seo Hyun Jeon et al. "Automotive Hardware Development According to ISO26262" International Conference on Advanced Communications Technology 2011.

[78]. ISO 26262-1:2011 "Road vehicles-Functional Safety". (www.iso.org)

[79]. H. Quinn, P. Graham, A. Manuzzato, T. Fairbanks, N. Dallmann, R. DesGeorges, "Neutron Sensitivity of High-Speed Networks", IEEE Transactions on Nuclear Science, Vol. 57, No. 6, pp. 3547-3552, December 2010.

[80]. R. Trautner, P. Armbruster, P. Fabry, A. Fernandez-Leon, J. Ilstad, D. Jameux, M. Suess, R. Wigand, "Avionics Architectures and Components for Planetary Entry Probe Payloads and Systems", 6th International Planetary Probe Workshop (IPPW6), Altanta, 2008.

[81]. E. Petri, S. Saponara, M. Tonarelli, I. Del Corona, L. Fanucci, P. Terreni, "Mitigating Radiation Effects on ICs at Device and Architectural Levels: the SpaceWire Router Case Study", IEEE International Symposium on Industrial Electronics (ISIE), 2007.

[82]. J. Perez, M. Sonza Reorda, M. Violante, "Dependability Analysis of CAN Networks: an emulation-based approach", 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 537-544, 2003.

[83]. M. Betoluzzo, G. Buja, "A high-performance application protocol for fault-tolerant CAN networks", IEEE International Symposium on Industrial Electronics, pp. 1705-1710, July 2010.

[84]. K. Echtle, "Fault Tolerant communication in safety relevant automotive applications", 20th International Conference on Architecture of Computing Systems, pp. 1-4, March 2007.

[85]. H. Kimm, H. Ham, "Integrated Fault Tolerant Systems for Automotive Bus Networks", 2nd International Conference on Computer Engineering and Applications, pp. 486-490, March 2010.

[86]. Y.K. Bong, J.K. Yun, "The dependability analysis of LIN network for adaptive front-lighting system", International SoC Design Conference, 2008, pp. I-425-I-428.

[87]. V.Kerzerho, H. G. Kerhoff, G-J. Bollen, Y. Xing, "The search for resilience weak spots in automotive mixed signal circuits", 17th IEEE International Mixed-Signals, Sensors and Systems Test Workshop, pp. 137-142, May 2011.

[88]. J. Stelzer, "LIN bus emerging standard for body control applications" EE Times ASIA , February 2004.

[89]. J. Suwatthikul, Book Chapter: "Fault detection and diagnosis for in-vehicle networks". Book: "Fault Detection", Wei Zhang (Ed.), ISBN: 978-953-307-037-7, InTech, Available from: http://www.intechopen.com/books/faultdetection/fault-detection-and-diagnosis-for-in-vehicle-networks

[90]. A. Cook, et al. "Reuse of Structural Volume Test Methods for In-System Testing of Automotive ASICs", 21st Asian Test Symposium, pp. 214-219, 2012.

[91]. H. Huangshui, Q. Guihe, "Online Fault Diagnosis for Controller Area Networks", International Conference on Intelligent Computation Technology and Automation, pp. 452-455, 2011.

[92]. S. Thanagasundram et al. "Failure Management for Reliable Automotive Start Up Process", IEEE International Conference on Computer Science and Automation Engineering, pp. 215-219, 2011.

[93]. Y. Sedaghat, S. G. Miremadi, "A Low-Cost On-Line Monitoring Mechanism for the FlexRay Communication Protocol", 4th Latin-American Symposium on Dependable Computing, pp. 111-118, 2009.

[94]. E. Armengaud et al. "Towards a Systematic Test for Embedded Automotive Communication Systems", IEEE Transactions on Industrial Informatics, Vol. 4, No, 3, August 2008.

[95]. M. Popa, V. Graz, A. Botas, "Lin Bus Testing Software", Canadian Conference on Electrical and Computer Engineering, pp. 1287-1290, 2006.

[96]. A. Vaskova et al. "Hardening of serial communication protocols for potentially critical systems in automotive applications: LIN bus", 19th IEEE International On-Line Testing Symposium, pp. 13-18, 2013.

[97]. A. Vaskova, M. Portela-García, M. García-Valderas, C. López-Ongil, M. Sonza Reorda "Permanent faults on LIN networks: On-line test generation". 176-181, IOLTS 2014.

[98]. P. Bernardi, M. Bonazza, E. Sanchez, M. Sonza Reorda, O. Ballan, "On-line functionally untestable fault identification in embedded processor cores", Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1462 – 1467, 2013.

[99]. C. Lopez-Ongil, M. Garcia-Valderas, M. Portela-Garcia, and L. Entrena, "Autonomous Fault Emulation: A New FPGA-Based Acceleration System for Hardness Evaluation," IEEE Trans. Nucl. Sci., vol. 54, no. 1, pp. 252–261, Feb. 2007.

[100]. J.M. Mogollón et al. "FTUnshades2: A Novel Platform for Early Evaluation of Robustness Against SEEs" Seville, Spain. RADECS 2011.

[101]. M. Alderighi et al. "Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform" 22 nd IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems. pp 105-113. 2007.

[102]. J.M. Mogollón, "Contributions to the Detection and Diagnosis of Soft Errors in Radiation Enviroments". Tesis Doctoral. 2013

[103]. Centro Nacional de Aceleradores(CNA), Sevilla, España.

[104]. Instituto Nacional de Técnica Aeroespacial (INTA). Madrid, España

[105]. M. Portela et al "Sensitivity Evaluation Method for Aerospace Digital Systems With Collaborative Hardening" IEEE T. on Nuclear Science. Vol 58. Iss.3. 2011.

[106]. A. Vaskova et al. "SEU Sensitivity Comparison for Different Reprogrammable Technologies with Minority Check Block" IEEE T. on Nuclear Science. Vol 60. Iss 4. 2013.

[107]. A. Vaskova, C. López-Ongil, M. García-Valderas, M. Portela-García, L. Entrena "Evaluation techniques for on-line testing of robust systems based on critical tasks distribution". IOLTS 2011: 258-263.

[108]. M. García-Valderas, M. Portela-García, C. López-Ongil, L. Entrena, A. Martin-Ortega, J. R. de Mingo, M. Álvarez, S. Esteve, S. Rodríguez. "The Effects of Proton Irradiation on CoolRunner-II CPLD Technology" Radiation Effects on Components and Systems Workshop, Finland 2008.

[109]. A. Vaskova,, et al. "Verifying Hardening Techniques for Distributed Electronic Systems in Critical Applications." Radiation and Its Effects on Components and Systems (RADECS), 2015 15th European Conference on. IEEE, 2015.

[110]. Felipe Restrepo-Calle, Sergio Cuenca-Asensi, Antonio Martínez-Álvarez, Fernanda Lima Kastensmidt "Considerations on application of selective hardening based on software fault tolerance techniques". LATS 2015.

[111]. A. Vaskova, A. Fabregat, M. Portela-García, M. García-Valderas, C. López-Ongil and M. S. Reorda, "Reducing SEU sensitivity in LIN networks: Selective and collaborative hardening techniques," 2014 15th Latin American Test Workshop - LATW, Fortaleza, 2014, pp. 1-6.

[112]. A. Vaskova, M. Portela-García, C. López-Ongil, E. Sánchez and M. S. Reorda, "About the functional test of permanent faults in distributed systems," Design of Circuits and Integrated Systems (DCIS), 2015 Conference on, Estoril, 2015, pp. 1-6.

[113]. C. López Ongil, M. Portela García "Assessing and Implementing the Fault Tolerance of Digital Circuits", BELAS School. Torino, June 1st 2016