

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un sistema de reconfiguración basado en un microcontrolador para un SoC FPGA en una carga útil bajo el estándar CubeSat

TESIS

Que para obtener el título de

Ingeniero Aeroespacial

P R E S E N T A

Cristian Alejandro Sánchez Calvillo

DIRECTOR DE TESIS

Ing. Israel Pérez Vicente



Ciudad Universitaria, Cd. Mx., 2026

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado DESARROLLO DE UN SISTEMA DE RECONFIGURACION BASADO EN UN MICROCONTROLADOR PARA UN SOC FPGA EN UNA CARGA UTIL BAJO EL ESTANDAR CUBESAT que presenté para obtener el titulo de INGENIERO AEROESPACIAL es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

CRISTIAN ALEJANDRO SANCHEZ CALVILLO
Número de cuenta: 318164474

Agradecimientos

A quienes estuvieron presentes,
incluso cuando el camino parecía incierto.

Deseo manifestar mi más profundo agradecimiento a mi familia, por haber sido parte fundamental en esta travesía. A mi hermana Saida y a mi hermano Said, por su apoyo, sus palabras de aliento y por impulsarme a seguir adelante. De manera especial, agradezco a mi madre, María, por creer en mí incluso cuando yo dejé de hacerlo, por no permitirme abandonar este proyecto y por recordarme que era capaz de concluirlo.

A mi asesor de tesis, el Ing. Israel Pérez Vicente, y al Doctor Saúl de la Rosa Nieves, por su guía, paciencia y acompañamiento durante el desarrollo de este trabajo. Gracias por orientarme en cada etapa de este viaje académico.

A Piña, por haber sido un amigo extraordinario y por impulsarme siempre a dar lo mejor de mí. A Zhara, Ericka, Pablo, Shevy y Alicia, por su apoyo, por los momentos compartidos y por hacer de esta etapa una parte inolvidable de mi vida.

Al laboratorio LIESE y a cada uno de sus integrantes, por brindarme un espacio en el que pude crecer académica, profesional y personalmente. Gracias por permitirme desarrollar proyectos significativos, aprender de grandes personas y compartir momentos que marcaron mi formación.

Por último, pero no menos importante, agradezco a Carlos, por haber sido un apoyo constante y un mentor desde el momento en que lo conocí. A la Arquitecta, por las lecciones compartidas, los momentos vividos y por haber dejado una huella especial en esta etapa de mi vida. A Rosalinda, por ser una persona extraordinaria, por sus consejos, sus enseñanzas y por impulsarme a crecer y mejorar como persona.

A todas las personas que formaron parte de este proceso, gracias por acompañarme, creer en mí y contribuir, de una u otra forma, a la culminación de este trabajo.

Resumen

Los dispositivos SoC FPGA representan una alternativa para aplicaciones espaciales debido a su capacidad de integrar procesamiento y lógica programable en un mismo dispositivo. Sin embargo, su funcionamiento depende de la información de configuración almacenada en la región de lógica programable, la cual puede verse afectada durante la operación del sistema. Ante esta situación, disponer de mecanismos externos de reconfiguración permite contar con una alternativa para restablecer la configuración del dispositivo cuando sea requerido.

En este trabajo se presenta el diseño e implementación de un sistema de reconfiguración externa para la región de lógica programable de un SoC FPGA *Xilinx Zynq*-7000 (XC7Z020). La propuesta emplea un microcontrolador MSP430FR5969 como elemento de control encargado de gestionar la información de configuración y realizar su transferencia hacia el SoC FPGA mediante una interfaz JTAG emulada por software. La arquitectura considera además una fuente de almacenamiento externa para resguardar el archivo de configuración requerido durante el proceso.

Como parte del desarrollo, se realizó el análisis de la estructura del *bitstream*, el diseño de los algoritmos necesarios para la comunicación mediante JTAG y la implementación del procedimiento de transferencia de la información de configuración. Asimismo, se desarrolló una herramienta auxiliar para facilitar el análisis e interpretación de los archivos utilizados durante el proceso.

Finalmente, el sistema fue implementado y verificado funcionalmente bajo condiciones de laboratorio. Los resultados obtenidos permitieron comprobar la comunicación entre el microcontrolador y el SoC FPGA, así como la transferencia del *bitstream* y la reconfiguración de la región de la lógica programable mediante el método propuesto.

Índice general

Agradecimientos **I**
Resumen **II**
Índice general **III**
Índice de figuras **VI**
Índice de tablas **VIII**
Siglas **IX**
1 Introducción **1**
1.1 Presentación del problema 2
1.2 Hipótesis 3
1.3 Objetivo 3
1.3.1 Objetivos específicos 3
1.4 Alcances 4
1.5 Justificación 4
2 Marco teórico **5**
2.1 *Bitstream* 5
2.1.1 Modalidades de configuración 6
2.1.2 Formatos de archivo de configuración 6
2.1.3 Modos de arranque y rutas de carga del *bitstream* 7
2.2 JTAG 10
2.2.1 Puerto TAP 11
2.2.2 Arquitectura de la lógica de prueba 11
2.3 SPI 22
2.3.1 Arquitectura y señales 22
2.3.2 Modos y parámetros de operación 24
2.3.3 Ventajas y limitaciones 25
2.4 Tecnologías de almacenamiento 26
2.4.1 Clasificación 26
2.4.2 Fundamentos físicos de las tecnologías de almacenamiento 27
2.4.3 Comparación tecnológica 34

3 Estado del arte **35**
3.1 Proyectos relacionados con la reconfiguración de dispositivos FPGA y SoC FPGA . . . 35
3.1.1 Reconfiguración de alta confiabilidad en órbita para FPGA tipo *flash* en una misión de espacio profundo . . . 36
3.1.2 Módulo de reconfiguración remota de FPGA para aplicaciones satelitales . . . 37
3.1.3 Reconfiguración de FPGA en operación para compresión hiperespectral flexible en un *CubeSat* . . . 38
3.1.4 DPR de coprocesadores para procesamiento de imágenes en satélites pequeños . . . 40
3.1.5 Unidad de procesamiento de datos reconfigurable tipo SoC para instrumentación espacial . . . 41
3.1.6 Plataforma SoC basada en FPGA para cómputo espacial reconfigurable . . . 43
3.1.7 Computadora central de a bordo reconfigurable implementada mediante FPGA basada en SRAM y *flash* para aplicaciones espaciales 44
3.2 Análisis comparativo de las soluciones reportadas y áreas de oportunidad . 46
4 Metodología de diseño del sistema de reconfiguración **49**
4.1 Metodología de diseño adoptada . . . 49
5 Diseño **51**
5.1 Objetivo del sistema de reconfiguración . . . 51
5.2 Diseño conceptual . . . 51
5.2.1 Descripción general del sistema de reconfiguración . . . 52
5.2.2 Requisitos del sistema de reconfiguración . . . 53
5.2.3 Herramienta auxiliar para el análisis del bitstream . . . 54
5.3 Diseño a nivel de sistema . . . 54
5.3.1 Arquitectura general del sistema de reconfiguración . . . 54
5.3.2 Identificación de subsistemas e interfaces de comunicación . . . 55
5.3.3 Diagrama de estados del sistema de reconfiguración . . . 58
5.4 Diseño de detalle . . . 59
5.4.1 Diseño del subsistema de almacenamiento . . . 60
5.4.2 Diseño del subsistema de control . . . 62
5.4.3 Diagrama de bloques del sistema de reconfiguración . . . 65
5.4.4 Consideraciones arquitectónicas del SoC FPGA . . . 66
5.4.5 Análisis del *bitstream* y herramienta de decodificación . . . 69
5.4.6 Diagrama de flujo del proceso de reconfiguración . . . 72
6 Pruebas y resultados **74**
6.1 Decodificador *bitstream* . . . 74
6.2 Emulación de JTAG . . . 77
6.3 Transferencia del archivo *bitstream* . . . 78
7 Conclusiones **81**
7.1 Trabajo a futuro . . . 82

Índice de figuras

2.1 Estructura general del estándar JTAG 10
2.2 Diagrama de estados del controlador TAP 12
2.3 Diagrama de estados del controlador TMP 14
2.4 Diagrama del registro de instrucciones 16
2.5 Estructura general de los registros de datos 19
2.6 Estructura general de SPI 22
2.7 Modos de operación de SPI 24
2.8 Celda de memoria SRAM 28
2.9 Celda de memoria DRAM 28
2.10 Celda de memoria EPROM 29
2.11 Celda de memoria EEPROM 30
2.12 Celda de memoria *flash* 30
2.13 Arquitecturas de memoria *flash* 31
2.14 Celda de memoria FRAM 32
2.15 Celda de memoria MRAM 33

4.1 Diagrama de metodología de diseño de Ulrich adaptada 50

5.1 Diagrama de bloques del sistema de reconfiguración 53
5.2 Tamaño del *bitstream* 55
5.3 Frecuencia máxima de TCK para dispositivos SoC FPGA de *Xilinx* 56
5.4 Diagrama de bloques del sistema de reconfiguración en la etapa de diseño a nivel de sistema 58
5.5 Diagrama de estados del sistema de reconfiguración. 59
5.6 Diagrama de bloques del sistema de reconfiguración 66
5.7 Elementos relevantes de la arquitectura interna del SoC FPGA 67
5.8 División de la arquitectura interna del SoC FPGA 68
5.9 Instrucciones empleadas para el acceso y control de las regiones PS y PL . 68
5.10 Patrones de detección del ancho del bus. Fuente: [19]. 69
5.11 Palabra de sincronización perteneciente a dispositivos de la serie 7 de *Xilinx*. Fuente: [19]. 69
5.12 Paquetes tipo uno. Fuente: [19]. 70
5.13 Paquetes tipo dos. Fuente: [19]. 70
5.14 Algunos de los registros de los paquetes tipo uno. Fuente: [19]. 71
5.15 Ejemplo de los elementos que conforman la estructura del archivo *bitstream* 71
5.16 Diagrama de flujo del procedimiento de reconfiguración del SoC FPGA. . . 72

6.1 Archivo *bitstream* de un proyecto en *Vivado*. 75
6.2 Función de búsqueda. 75
6.3 Diccionario de acciones por registro identificado 76

6.4 Archivo *bitstream* del *blink* decodificado. 76
6.5 Archivo *bitstream* del *blink* decodificado. 77
6.6 Prueba de comunicación exitosa 78
6.7 Transferencia exitosa del archivo *bitstream*. 79
6.8 Transferencia del archivo *bitstream* 79
6.9 Desplazamiento de estados y envío de instrucciones 80
6.10 Transferencia del *bitstream* en bytes 80

Índice de tablas

2.1 Comparación tecnológica de memorias 34

3.1 Comparación de soluciones reportadas y áreas de oportunidad 47

5.1 Comparación de memorias no volátiles 61
5.2 Comparación general entre microcontroladores y dispositivos PLD 63
5.3 Análisis comparativo de alternativas de microcontroladores 64

Siglas

AMD	*Advanced Micro Devices*
ARM	*Advanced RISC Machine*
BIN	*Binary Bitstream File*
BIT	*Bitstream File*
BSC	*Boundary Scan-Cell*
BSDL	*Boundary Scan Description Language*
CMOS	*Complementary Metal Oxide Semiconductor*
COTS	*Commercial off-the-shelf*
CPHA	*Clock Phase*
CPOL	*Clock Polarity*
CRAM	*Configuration Random Access Memory*
CS/SS	*Chip Select/Signal Select*
DAP	*Debug Access Port*
DevC	*Device Configuration*
DPR	*Dynamic Partial Reconfiguration*
DPU	*Data Processing Unit*
DR	*Data Register*
DRAM	*Dynamic Random Access Memory*
DRO	*Distant Retrograde Orbit*
ECID	*Electronic Chip Identification*
EEPROM	*Electrically Erasable Programmable Read Only Memory*
EPROM	*Erasable Programmable Read Only Memory*
FPGA	*Field Programmable Gate Array*
FRAM	*Ferroelectric Random Access Memory*
FSR	*Full Static Reconfiguration*

GPIO	*General Purpose Input/Output*
HEX	*Hexadecimal Configuration File*
HZO	*Hafnium Zirconium Oxide*
ICAP	*Internal Configuration Access Port*
IEEE	*Institute of Electrical and Electronics Engineers*
IIC/I2C	*Inter Integrated Circuit*
IR	*Instruction Register*
JTAG	*Joint Test Action Group*
LETh	*Linear Energy Transfer*
LSB	*Least Significant Bit*
MB	*Mega Byte*
Mb	*Mega bit*
MCS	*Memory Configuration Store*
MISO	*Master Input - Slave Output*
MOSFET	*Metal-Oxide-Semiconductor Field-Effect Transistor*
MOSI	*Master Output - Slave Input*
MRAM	*Magnetoresistive Random Access Memory*
MSB	*Most Significant Bit*
NAND	*Not And*
NASA	*National Aeronautics and Space Administration*
NOR	*Not Or*
OBC	*On-Board Computing*
OPU	*On-board Processing Unit*
PCAP	*Processor Configuration Access Port*
PL	*Programmable Logic*
PLD	*Programmable Logic Device*
PS	*Processing System*
PZT	*Lead Zirconate Titanate*
QSPI	*Quad Serial Peripheral Interface*
RBT	*Raw Bitstream Text*
RDM	*Radiation Design Margin*
RGB	*RED, GREEN, BLUE*

RPD	*Raw Programming Data*
SCMI	*Sistema de Control y Manejo de la Información*
SCLK	*Serial Clock*
SD	*Secure Digital*
SEE	*Single Event Effect*
SEB	*Single Evento Burnout*
SEFI	*Single Event Functional Interrupt*
SEGR	*Single Event Gate Rupture*
SEL	*Single Evento Latchup*
SET	*Single Event Transient*
SEU	*Single Event Upset*
SMC	*Static Memory Controller*
SoC	*System on Chip*
SoC FPGA	*System on Chip Field Programmable Gate Array*
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
SSD	*Solid State Driver*
STT	*Spin-Transfer Torque*
TAP	*Test Access Port*
TCK	*Test Clock*
TDI	*Test Data In*
TDO	*Test Data Out*
TID	*Total Ionizing Dose*
TMR	*Triple Modular Redundancy*
TMP	*Test Mode Persistence*
TMS	*Test Mode Select*
TRST	*Test Reset*
UART	*Universal Asynchronous Receiver-Transmitter*
VMC	*Visual Monitoring Camera*
WSP	*Warm Standby Protection*

Capítulo 1

Introducción

Los satélites tipo *CubeSat*, desde su creación en 1999 por los profesores Jordi Puig-Suari y Bob Twiggs, surgieron con el propósito de otorgar acceso al espacio a la comunidad universitaria. Con el tiempo, adquirieron gran relevancia debido a su bajo costo y al tiempo reducido necesario para su desarrollo. Esto permitió que las comunidades universitarias desarrollaran tecnología espacial. Actualmente, los satélites *CubeSat* son empleados en proyectos del sector espacial con propósitos de brindar servicios en telecomunicaciones, estudios del planeta y seguridad nacional, entre otros [1, 2].

Al igual que cualquier satélite, los *CubeSat* están integrados por sistemas que trabajan en conjunto para llevar a cabo los objetivos de la misión, entre los cuales destaca el sistema de control y manejo de la información (C&DH, por sus siglas en inglés), encargado de ejecutar los comandos para controlar los sistemas y monitorear el estado del satélite en tiempo real; además, gestiona la información que se almacena, procesa y transmite durante toda la misión [3]. Debido a la relevancia de estas funciones, la presencia de una falla en el satélite puede comprometer su funcionamiento esperado. En este contexto, de acuerdo con [4] y [5], una falla se entiende como una anomalía o un comportamiento que puede provocar uno o varios errores y afectar el funcionamiento esperado del sistema, ya sea en el *hardware* o *software*. Por lo tanto, cualquier falla en el funcionamiento de algún sistema, especialmente en el C&DH, puede repercutir directamente en el funcionamiento del satélite.

En un entorno hostil como el espacio, una de las principales causas de fallas en los sistemas electrónicos es la radiación espacial, compuesta principalmente de radiación cósmica y partículas atrapadas en los cinturones de radiación. La exposición a este ambiente puede provocar efectos significativos en el funcionamiento de los sistemas electrónicos de cualquier satélite, provocando alteraciones temporales o permanentes en los circuitos integrados [6, 7].

Debido a lo anterior, en el diseño de sistemas electrónicos destinados a aplicaciones espaciales es necesario considerar los efectos producidos por la radiación sobre los dipositivos semiconductores. Estos efectos pueden clasificarse en dos categorías principales: efectos acumulativos y efectos de evento único [8, 9, 10].

La TID corresponde a un efecto acumulativo producido por la exposición del dispositivo a radiación ionizante. Durante esta interacción se generan pares electrón-hueco en los materiales del semiconductor; mientras los electrones presentan una mayor movilidad, parte de los huecos puede quedar atrapada principalmente en regiones de óxido. La acumulación de esta carga puede modificar las características eléctricas de los transistores y, conforme aumenta la dosis absorbida, provocar una degradación en el comportamiento del circuito [8, 9, 10].

El DD también constituye un efecto acumulativo, pero se origina por la interacción de partículas energéticas con los átomos de la red cristalina del semiconductor. Cuando la energía transferida es suficiente, un átomo puede ser desplazado de su posición, generando defectos en la estructura del material. Estos defectos pueden actuar como centros de atrapamiento o recombinación de portadores y modificar las propiedades eléctricas del dispositivo. Este fenómeno se encuentra asociado principalmente con colisiones producidas por partículas como protones y neutrones [10].

A diferencia de los anteriores, los SEE corresponden a efectos de evento único y son provocados por la interacción de una sola partícula energética con una región sensible del dispositivo. El paso de la partícula puede generar carga eléctrica que, al ser recolectada por los nodos del circuito, produce una perturbación en su funcionamiento. Dependiendo de sus consecuencias, los SEE pueden clasificarse en destructivos y no destructivos. Entre los efectos no destructivos se encuentran el transitorio por evento único (SET, por sus siglas en inglés), la perturbación por evento único (SEU, por sus siglas en inglés) y la interrupción funcional por evento único (SEFI, por sus siglas en inglés). Por otro lado, entre los efectos destructivos se encuentran el enclavamiento por evento único (SEL, por sus siglas en inglés), el quemado por evento único (SEB, por sus siglas en inglés) y la ruptura de compuerta por evento único (SEGR, por sus siglas en inglés) [8, 9, 10].

Por esta razón, surge la necesidad de implementar sistemas tolerantes a fallas, capaces de detectar y aislar anomalías, así como de mitigar sus efectos y, cuando sea posible, recuperar el funcionamiento del sistema, con el propósito de aumentar las probabilidades de éxito de la misión [11]. Para ello, existen diversas estrategias para implementar técnicas tolerantes a fallas: el enmascaramiento de fallas, mecanismos de detección, recuperación de fallas y reconfiguración. En el caso de aplicaciones espaciales, la reconfiguración cobra importancia debido a la necesidad de recuperar la funcionalidad del sistema ante determinadas fallas o de modificar su configuración de acuerdo con los requisitos de la misión [12, 13, 14]. Ante este escenario la presente tesis propone el diseño de un sistema de reconfiguración para un dispositivo maestro SoC *Xilinx Zynq*-7000 (XC7Z020) implementado en un dispositivo reconfigurador. El propósito de este sistema consiste en contribuir a mantener el funcionamiento del dispositivo maestro y permitir su adaptación a diferentes objetivos de la misión, aún cuando sufra efectos debidos a la radiación espacial.

1.1. Presentación del problema

En los proyectos espaciales basados en satélites tipo *CubeSats*, especialmente en aquellos que requieren mayor capacidad de procesamiento o interfaces avanzadas, es común el

uso de dispositivos lógicos programables (PLD, por sus siglas en inglés), tales como los arreglos de compuertas programables en campo (FPGA, por sus siglas en inglés) y los arreglos de compuertas programables en campo con sistema en chip integrado (SoC FPGA, por sus siglas en inglés). De acuerdo con [15], estas plataformas se emplean cuando la misión demanda procesamiento acelerado gracias a su flexibilidad y alto desempeño. Sin embargo, debido a que estos dispositivos dependen de un archivo de configuración conocido como *bitstream* para definir su lógica interna, si este archivo se altera o la configuración interna actual del dispositivo se modifica debido a fallas provocadas por la radiación, el dispositivo puede quedar en un estado indeterminado que le impida operar correctamente.

En proyectos espaciales de bajo presupuesto es común emplear componentes conocidos como componentes COTS. No obstante, el hecho de pertenecer a esta categoría no implica necesariamente que presenten una baja tolerancia a la radiación, sino que, en general, no cuentan con una calificación específica para aplicaciones espaciales ni con un proceso de aseguramiento de dureza frente a radiación que garantice su comportamiento bajo este tipo de ambiente. Por esta razón, su respuesta frente a los efectos de la radiación debe evaluarse a partir de datos experimentales, ensayos reportados o caracterizaciones especificas del dispositivo. En ausencia de esta información, no es posible determinar un comportamiento aproximado dentro de un entorno espacial. Como consecuencia, resulta necesario considerar mecanismos de recuperación y reconfiguración que permitan restablecer la operación del sistema ante posibles alteraciones en su funcionamiento [16]. Para ello, se plantea el diseño de un sistema capaz de restaurar la configuración del dispositivo maestro mediante la carga del *bitstream* correspondiente a través de un dispositivo reconfigurador.

1.2. Hipótesis

Se puede desarrollar un sistema reconfigurador que, por medio de la interfaz grupo de acción conjunta para pruebas (JTAG, por sus siglas en inglés), permita reconfigurar la lógica programable (PL, por sus siglas en inglés) de un SoC FPGA a bordo de un *CubeSat*.

1.3. Objetivo

Diseñar un sistema de reconfiguración basado en un microcontrolador MSP430FR5969 para un SoC FPGA *Xilinx Zynq*-7000 (XC7Z020), encargado de gestionar la recepción del archivo de configuración desde una fuente de almacenamiento y su transmisión hacia el dispositivo maestro a través de JTAG emulado.

1.3.1. Objetivos específicos

1. Diseño del algoritmo y su implementación en *software* para realizar la decodificación del archivo de configuración *bitstream*.

2. Diseño del algoritmo y su implementación en *software* capaz de emular JTAG.
3. Diseño del algoritmo y su implementación en *software* para realizar la transferencia del archivo *bitstream*.
4. Realizar el diseño de detalle del sistema de reconfiguración basado en un dispositivo de control, una fuente de almacenamiento y el SoC FPGA *Xilinx Zynq*-7000 (XC7Z020).

1.4. Alcances

El desarrollo de un sistema de reconfiguración para un SoC FPGA requiere de delimitar con claridad los alcances técnicos. A continuación, se presentan los alcances del presente trabajo.

- El proyecto se enfoca en el diseño, implementación y verificación funcional de un sistema de reconfiguración externa para el SoC FPGA *Xilinx Zynq*-7000 (XC7Z020), por medio de un dispositivo de control.
- La verificación funcional del sistema se realiza bajo condiciones de un laboratorio, en donde la integración de un *CubeSat* se contempla como un trabajo a futuro.

1.5. Justificación

El uso de dispositivos con PL, como los SoC FPGA, en plataformas *CubeSat* permiten integrar funciones de control, adquisición y procesamiento de datos en un único componente [15]. Combina lógica programable con un sistema de procesamiento, lo que lo convierte en una buena opción para plataformas limitadas de volumen y área como los satélites *CubeSat* [17]. No obstante, la operación de la región PL requiere de un archivo de configuración, por lo que cualquier falla que altere este archivo puede dejar al sistema en un estado de operación indeterminado.

En proyectos de *CubeSat* de bajo presupuesto se emplean componentes COTS, los cuales pueden ser susceptibles a efectos inducidos por radiación, por lo que se deben de considerar las características del dispositivo y las condiciones de la misión. En el caso de los SoC FPGA, es indispensable el desarrollo de una herramienta de reconfiguración basada en JTAG con el fin de que estos sistemas sean viables en aplicaciones espaciales.

Capítulo 2

Marco teórico

El propósito de este capítulo es presentar y explicar las bases conceptuales y proporcionar el contexto teórico necesario para comprender la problemática y la solución propuesta. En primer lugar, se introduce el concepto de *bitstream*, pieza fundamental en la configuración interna de los PLD, en este caso, de los FPGA y de la región PL de los SoC FPGA. Posteriormente, se aborda el estándar IEEE 1149.1, conocido comúnmente como JTAG, mediante la descripción de su arquitectura, los distintos controladores, registros e instrucciones que permiten realizar pruebas de lógica interna, depuración y transmisión de información binaria. Después, se detalla la interfaz periférica serial (SPI, por sus siglas en inglés), una interfaz de comunicación síncrona que permite el intercambio de datos en modo *full-duplex* mediante líneas independientes para la transmisión y recepción de información, donde se describen sus características técnicas, modos de operación y arquitectura interna. Finalmente, se presentan tecnologías de almacenamiento de información, lo que incluye memorias volátiles y no volátiles, sus mecanismos de almacenamiento y un análisis comparativo, con especial énfasis en su tolerancia a la radiación. En conjunto, estos temas conforman el márco teórico que sustenta el diseño y la implementación del proyecto.

2.1. *Bitstream*

El *bitstream* es el archivo de configuración que contiene la información binaria necesaria, desde comandos y datos, para configurar el *hardware* interno de un FPGA o de la región PL de un SoC FPGA; es decir, describe cómo deben inicializarse y conectarse los recursos lógicos internos, así se determina el comportamiento definido por el diseñador. Este archivo se genera al finalizar el flujo de implementación y contiene los datos necesarios para configurar los recursos programables y las interconexiones del dispositivo de acuerdo con el diseño realizado [18, 19, 20].

Se requiere de una interfaz especializada para la transmisión de este archivo, por ejemplo JTAG, herramienta utilizada para poder cargar el archivo a la memoria interna del dispositivo. Asimismo, cada modelo de FPGA cuenta con características específicas, por lo tanto, requieren de un *bitstream* único para sus respectivas configuraciones internas [19].

2.1.1. Modalidades de configuración

Dentro de los esquemas de configuración de la lógica programable empleados en una FPGA o SoC FPGA, existen dos modalidades de configuración: *full-bitstream* y *partial-bitstream*. La elección de uno de ellos depende de la necesidad de configurar en su totalidad el dispositivo o solo ciertas zonas específicas dentro del mismo. A continuación se presentan ambas modalidades de configuración:

- *Full-bitstream*: de acuerdo con [18] y [20], un *full-bitstream* hace referencia al archivo que contiene los datos de configuración completa del dispositivo.
- *Partial-bitstream*: se trata de una configuración parcial del dispositivo; es decir, contiene únicamente los *frames* asociados a una región específica, el resto permanece sin modificaciones [21].

2.1.2. Formatos de archivo de configuración

De acuerdo con [19], cada proveedor define sus propios formatos, estructuras, encabezados, método de compresión y mecanismos de carga, todo de acuerdo con la arquitectura interna de cada modelo. Sin embargo, a pesar de que todos contienen la información de configuración del dispositivo, no todos son cargados de forma directa al FPGA o SoC FPGA, ya que cada uno cumple con una función específica dentro del flujo de diseño y configuración.

En el caso de *Xilinx*, fabricante del dispositivo empleado en este trabajo, los formatos que maneja para su archivo de configuración, particularmente para los dispositivos pertenecientes a la familia SoC FPGA *Xilinx* Zynq-7000, son los siguientes:

- Archivo de *bitstream* (BIT, por sus siglas en inglés): formato estándar utilizado para la configuración del dispositivo desde herramientas oficiales de *Xilinx*, mediante interfaces como JTAG. Este archivo cuenta con la información binaria de configuración del dispositivo para su transmisión y carga directa, por lo que es uno de los formatos más utilizados para configurar los dispositivos. Además, cuenta con un encabezado que proporciona información general sobre el proyecto y el dispositivo de interés.
- Archivo de texto de *bitstream* (RBT, por sus siglas en inglés): Es el archivo de configuración representado en un formato de texto ASCII, en el cual cada bit de configuración se expresa como un carácter 0 o 1. Este formato no está orientado a la

programación del dispositivo, más bien es empleado en el análisis de la información de configuración, ya que permite inspeccionar el contenido del archivo mediante herramientas de *software* con fines de estudio.

- Archivo binario de *bitstream* (BIN, por sus siglas en inglés): archivo de configuración que contiene la información binaria cruda del *bitstream*; es decir, solo son unos y ceros, sin encabezados que proporcionen información acerca del proyecto. Este tipo de archivos resulta útil para procesos de configuración personalizados, donde un agente externo como un microcontrolador, es el encargado de transmisitir la información de configuración mediante un método compatible con el dispositivo.
- Archivo de almacenamiento de configuración de memoria (MCS, por sus siglas en inglés): formato del archivo orientado a la programación de memorias externas no volátiles como memorias *flash*. Básicamente es un archivo donde la información binaria se organiza de manera adecuada para su almacenamiento en una memoria externa, lo que permite cargar su configuración desde la memoria durante el proceso de arranque y así se le otorga configuración autónoma al sistema cada vez que se enciende.
- Archivo de configuración hexadecimal (HEX, por sus siglas en inglés): archivo de configuración que muestra el contenido del *bitstream* en formato hexadecimal, expresando grandes cantidades de información binaria de forma más legible y manejable. Representa un formato intermedio en flujos de trabajo personalizados. Principalmente se emplea cuando el *bitstream* debe integrarse con sistemas o herramientas que operan con datos hexadecimales, como memorias externas o microcontroladores.

Estos tipos de archivo se generan mediante herramientas especializadas de implementación y configuración durante la etapa final de diseño, en este caso desde el *software* de Vivado. La existencia de múltiples formatos de archivo significa que cada uno de ellos es usado en diferentes mecanismos de configuración, tales como: JTAG, interfaz serial periférica cuádruple (QSPI, por sus siglas en inglés) y tarjetas de memoria (SD, por sus siglas en inglés), entre otros. Debido a que en este trabajo se emplea un dispositivo perteneciente a *Xilinx*, solo se mencionan los formatos de este fabricante dentro de la familia SoC FPGA *Xilinx Zynq*-7000.

2.1.3. Modos de arranque y rutas de carga del *bitstream*

Los PLD se caracterizan por su versatilidad, no solo en la manera en como implementan funcionalidad de *hardware*, sino también en los diferentes mecanismos que proporcionan para iniciar su operación y cargar el *bitstream*. Estas dos características están en función de los objetivos actuales del proyecto y del entorno de aplicación. Esa misma regla aplica para los dispositivos SoC FPGA; es decir, los métodos de arranque determinan el medio desde el cual se obtiene el *bitstream*, así como las interfaces internas específicas que establecen la ruta mediante la cual el archivo es transferido a la región PL [17].

2.1.3.1. Modos de arranque

Como se mencionó, es importante seleccionar un modo de arranque adecuado para el desarrollo del proyecto, ya que este influye tanto en el tiempo de configuración del sistema como en la forma en que el dispositivo obtiene el *bitstream* en la fase de inicialización. Para los FPGA de la serie 7, la carga del *bitstream* puede realizarse mediante distintos modos de configuración, los cuales definen la interfaz utilizada y el origen de los datos de configuración. Por otro lado, los SoC FPGA también incorporan múltiples modos de arranque; sin embargo, a diferencia de los FPGA, estos tienen embebido una región de sistema de procesamiento (PS, por sus siglas en inglés), lo que hace que presenten diferencias adicionales en su arquitectura interna y, por lo tanto, el proceso de obtención y transferencia del *bitstream* a la región PL forman parte de un proceso más amplio, gestionado por el fabricante.

A continuación, se presentan los modos de arranque disponibles dentro de los dispositivos pertenecientes a la familia SoC FPGA *Xilinx Zynq*-7000.

- Modo de arranque JTAG: método de arranque en el que el dispositivo activa la interfaz JTAG para que un agente externo cargue el *bitstream* a la región PL en la memoria de configuración de acceso aleatorio (CRAM, por sus siglas en inglés). Este modo es comúnmente empleado en las etapas de desarrollo, depuración y validación del diseño porque permite la configuración del dispositivo sin depender del uso de memorias no volátiles externas ni de imágenes de arranque completas.
- Modo de arranque NOR: método de arranque en el cual el dispositivo accede a una memoria NOR *flash* externa asíncrona, conectada a un controlador de memoria estática (SMC, por sus siglas en inglés). En este método de arranque, la región PS obtiene la información de configuración almacenada en la memoria, la cual puede incluir el *bitstream*, y la tranfiere a la región PL como parte del proceso de inicialización del dispositivo.
- Modo de arranque NAND: método de arranque por el cual el dispositivo obtiene la información de configuración desde una memoria NAND *flash* externa, conectada al SMC. Al igual que el anterior método, la región PS es quien obtiene la información necesaria de la memoria externa para la inicialización del dispositivo, el cual puede contener el *bitstream* para la configuración de la región PL. Es capaz de soportar memorias NAND *flash* con un bus de datos de 8 y 16 bits. Este tipo de memorias es utilizada por su capacidad de almacenamiento.
- Modo de arranque Quad-SPI: método de arranque en el que el dispositivo obtiene la información de configuración desde una memoria *flash* externa Quad-SPI, a través de la región PS. Por medio de esta interfaz, el sistema obtiene la información de configuración almacenada en la memoria, la cual puede incluir el *bitstream* que posteriormente se transfiere a la región PL. El uso de la interfaz Quad-SPI, por medio de distintos anchos de bus, permite optimizar el tiempo de arranque del sistema. Este tipo de memoria es utilizada por su velocidad de acceso y su simplicidad.
- Modo de arranque SD Card: método de arranque en el cual el dispositivo obtiene la información de configuración desde una tarjeta de memoria perteneciente a la familia

SD, a través de la región PS. La información almacenada en la tarjeta, que puede incluir el *bitstream*, es leída por el PS y usada durante la inicialización del sistema. A diferencia de los otros métodos, emplear una memoria de este tipo proporciona una capacidad de almacenamiento considerablemente grande; además de su fácil gestión de archivos por medio de sistemas de archivos estándar como el formato FATS16/32 que resulta viable durante las etapas de desarrollo y pruebas.

2.1.3.2. Rutas de carga de *bitstream*

Así como existen diversos métodos de arranque del sistema, los dispositivos SoC FPGA, también cuentan con diversas formas para cargar el *bitstream* hacia la región PL. La selección de la ruta depende de los objetivos del proyecto y la etapa en la que se encuentre, ya sea durante pruebas rápidas, tareas de depuración, etc. A continuación, se describen de manera general las tres rutas de carga de *bitstream* disponibles en la familia SoC FPGA *Xilinx Zynq*-7000.

- JTAG: ruta de carga de *bitstream* por el cual un agente externo transfiere el archivo a la región PL a través del controlador conocido como puerto de acceso de prueba (TAP, por sus siglas en inglés). Esta ruta es normalmente utilizada durante etapas de desarrollo y pruebas, ya que permite configurar el dispositivo sin depender de la región PS ni de memorias externas.
- Puerto de acceso a la configuración del procesador (PCAP, por sus siglas en inglés): en esta ruta de configuración la región PS actúa como intermediario para transmitir el *bitstream* a la región PL. Por este método, el PS obtiene el archivo de configuración desde una memoria externa, puede ser una memoria QSPI-flash, NAND, NOR o SD. Posteriormente, lo envía a la región PL a través de la interfaz de configuración del dispositivo (DevC, por sus siglas en inglés).
- Puerto de acceso a la configuración interna (ICAP, por sus siglas en inglés): esta ruta de configuración solo permite configurar la región PL a través de lógica previamente implementada. Es comúnmente usado para aplicaciones de reconfiguración dinámica parcial, donde el *bitstream* es gestionado únicamente desde el interior de la región PL.

2.2. Interfaz de grupo de acción conjunta (JTAG)

Según [23, 24], anteriormente, la verificación de circuitos integrados se realizaba por medio de técnicas basadas en acceso físico, como *bed of nails*, las cuales requerían de contacto directo con los pines del dispositivo para realizar las pruebas de las señales internas. No obstante, el avance de la tecnología de miniaturización, el incremento de la densidad de pines, el uso de tecnologías de montaje superficial (SMT, por sus siglas en inglés) y la creciente complejidad de los circuitos integrados, entre otros factores, provocaron que este tipo de técnicas se volvieran poco eficientes o imprácticas [25]. Para darle solución a este problema de verificación y para contar con un método estandarizado que permitiera realizar las pruebas sin depender del acceso físico directo, a mediados de 1980, el grupo *Joint Test Action Group* desarrolló *Standard Test Access Port and Boundary-Scan Architecture*, comúnmente conocido como JTAG y tiempo después se formalizó en IEEE como el estándar 1149.1.

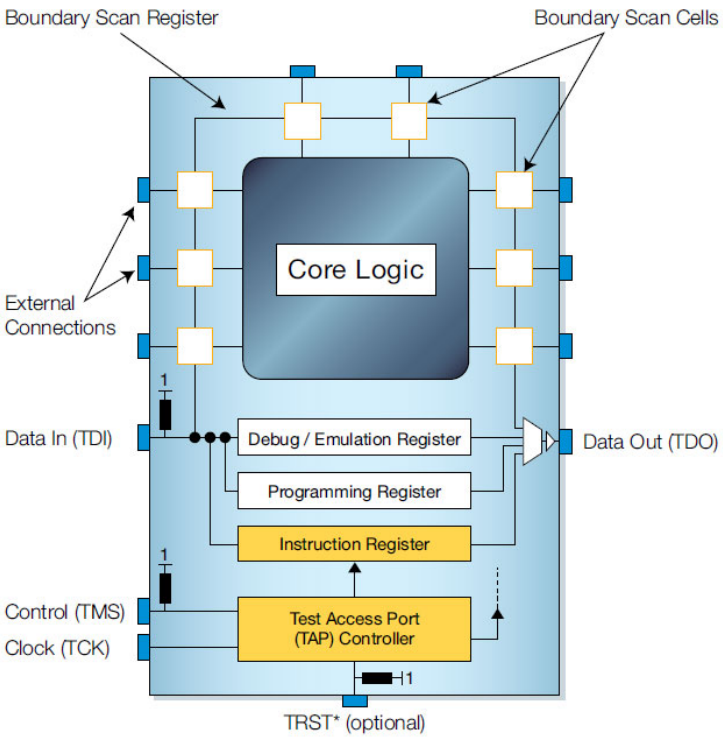


Figura 2.1: Estructura general del estándar JTAG. Fuente: [22].

El interfaz JTAG está compuesta por cinco señales: reloj de prueba (TCK, por sus siglas en inglés), selección del modo de prueba (TMS, por sus siglas en inglés), entrada de datos de prueba (TDI, por sus siglas en inglés), salida de datos de prueba (TDO, por sus siglas en inglés) y reinicio de prueba (TRST*, por sus siglas en inglés), conectadas a una máquina de estados interna denominada controlador TAP, la cual rige el acceso a distintos registros dentro del dispositivo compatible con JTAG y las operaciones que puede realizar. Cuenta con un protocolo de comunicación para escaneo de límites (*boundary-scan*), una técnica de pruebas que consiste en insertar un conjunto de celdas de escaneo de límites (BSC, por sus siglas en inglés) entre la lógica interna del circuito integrado y sus pines de entrada y salida. En conjunto, estas celdas forman el registro de escaneo de límites (BSR, por sus siglas en inglés), el cual permite observar y controlar el estado lógico de cada pin sin interferir directamente con la lógica funcional. Mediante esta técnica, es posible verificar las interconexiones entre dispositivos, detectar fallas de soldadura, cortocircuitos o pines abiertos, aislar la lógica interna del exterior durante procesos de pruebas y depuración, todo esto sin requerir de un acceso físico al encapsulado. La figura 2.1 muestra la estructura general de la interfaz JTAG, lo que incluye la lógica central del dispositivo, las señales involucradas, el controlador TAP, el BSR, los registros de datos y de instrucciones.

2.2.1. Puerto TAP

El puerto TAP es la interfaz física definida por el estándar 1149.1, por medio del cual se da acceso al sistema de prueba y depuración dentro de un circuito integrado. Está compuesto por cuatro líneas de comunicación y sincronización, y una quinta línea opcional de reinicio. A través de este puerto se transfieren los datos de pruebas e instrucciones a ejecutar en el dispositivo JTAG. La figura 2.1 muestra las conexiones correspondientes a estas señales. A continuación se describen las señales que lo componen.

- TCK: proporciona el reloj maestro para la sincronización de las operaciones y datos. La señal opera con un ciclo de trabajo del 50 %; sin embargo, puede mantenerse en nivel lógico bajo como en nivel lógico alto (opcional), así se mantienen los estados de los registros o se detienen momentáneamente las operaciones del proceso sin afectar el funcionamiento del protocolo.
- TMS: controla la máquina de estados del controlador TAP. El valor presente en la señal determina el estado siguiente dentro del diagrama de estados del controlador TAP y es muestreado en el flanco de subida de TCK, por lo que en el flanco de bajada su valor debe cambiar para garantizar estabilidad en el próximo muestreo.
- TDI: ingresa los datos e instrucciones de prueba a través de la cadena JTAG. Se muestrea en el flanco de subida de TCK, por lo que se espera que el cambio de dato se realice en el flanco de bajada de TCK para garantizar estabilidad al momento de capturar su valor. La información introducida por TDI es desplazada n ciclos de reloj hasta aparecer en la salida, donde n corresponde al número de bits del registro activo.
- TDO: es la señal que representa la salida de los datos e instrucciones de prueba a través de la cadena JTAG. La captura de los datos se realiza en el flanco de bajada de TCK. Mientras no se encuentre activa, tiene que permanecer en un estado inactivo o de alta impedancia, esto se logra al conectar un *buffer* triestado o un *flip-flop* (por lo general tipo D) sincronizado por flancos de bajada para evitar conflictos en la transmisión de datos, en especial en conexiones con múltiples dispositivos JTAG.
- $\overline{\text{TRST}}$: es una señal opcional que proporciona un reinicio asíncrono del controlador TAP y del controlador de persistencia del modo de prueba (TMP, por sus siglas en inglés). Su activación permite llevar la lógica de prueba a un estado conocido de reinicio, independientemente del estado en el que se encuentre el controlador.

2.2.2. Arquitectura de la lógica de prueba

La arquitectura JTAG incorpora una cadena *boundary-scan* y un conjunto de módulos dedicados al control, la gestión de instrucciones y al manejo de datos dentro de los dispositivos JTAG. Esta arquitectura está compuesta de elementos principales tales como los controladores de la lógica de prueba, el registro de instrucciones, las instrucciones y los registros de datos.

Esta configuración permite examinar la lógica en modo de prueba sin alterar el funcionamiento normal del chip, de modo que se forma una cadena de escaneo interconectada que proporciona la posibilidad de depurar, validar y diagnosticar el chip de manera eficiente.

2.2.2.1. Controladores de lógica de prueba

Los controladores de lógica de prueba son máquinas de estados finitos que controlan la secuencia de los estados del circuito. Existen dos máquinas de estados finitos dentro de la interfaz JTAG los cuales son:

Controlador TAP

Encargada de controlar el comportamiento de la lógica de prueba. Responde a las señales TMS, TCK y $\overline{\text{TRST}}$ para determinar el estado siguiente y mantener la sincronización en todos los componentes del sistema. El controlador TAP obedece al diagrama de estados mostrado en la figura 2.2 para realizar la gestión de instrucciones y datos. El valor que aparece junto a cada línea de transición de estados representa el nivel lógico presente en TMS, y es evaluado en cada flanco de subida de TCK para cambiar de estado.

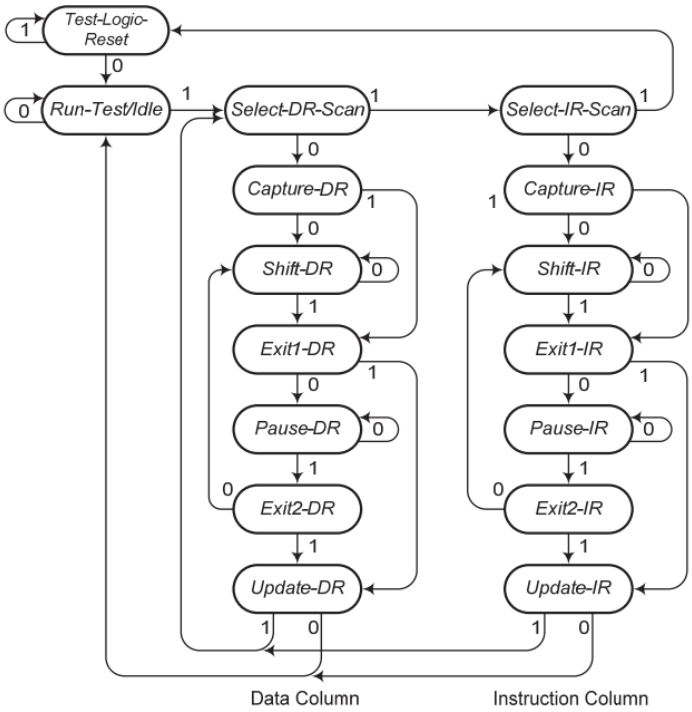


Figura 2.2: Diagrama de estados del controlador TAP.
Fuente: [24].

Los estados visualizados en el diagrama 2.2 se describen brevemente a continuación:

- *Test-Logic-Reset*: en este estado no se ejecuta ninguna acción de lógica de prueba; sin embargo, es posible realizar acciones preconfiguradas sin alterar este estado, como devolver el identificador del dispositivo o activar el registro *BYPASS*. Para entrar en

este estado, TMS debe mantener un nivel lógico alto durante cinco pulsos de reloj. Alternativamente, si el dispositivo dispone de ella, también puede emplearse $\overline{\text{TRST}}$ en nivel lógico bajo.

- *Run-Test/Idle*: en este estado se pueden ejecutar pruebas internas de la lógica del sistema, o bien mantener al sistema en espera sin realizar cambios en los valores de los registros.
- *Select-DR-Scan*: estado transitorio donde se define qué se opera con algún registro de datos.
- *Select-IR-Scan*: estado intermedio donde se define qué se opera con el registro de instrucciones.
- *Capture-DR*: estado en el que se cargan los datos de forma paralela al registro de datos previamente seleccionado por la instrucción activa. La captura se realiza en los flancos de subida de TCK.
- *Shift-DR*: se selecciona el registro de datos activo, conectado entre TDI y TDO, y desplaza bit a bit la información en cada flanco de subida de TCK.
- *Exit1-DR*: estado transitorio que indica el fin del desplazamiento de bits en el registro de datos.
- *Pause-DR*: en este estado se detiene temporalmente el desplazamiento de datos en el registro de datos en serie conectado entre TDI y TDO.
- *Exit2-DR*: es un estado intermedio que finaliza el desplazamiento de bits en el registro de datos activo, si previamente el controlador TAP se encontraba en el estado *Pause-DR*.
- *Update-DR*: los bits desplazados a través del registro de datos son cargados a la salida paralela del sistema. En este estado, se hace efectivo el cambio en el *hardware*.
- *Capture-IR*: en este estado, se cargan valores predefinidos al registro de instrucciones de forma paralela.
- *Shift-IR*: permite desplazar bit a bit la instrucción cargada en el registro de instrucciones.
- *Exit1-IR*: estado intermedio que finaliza el desplazamiento de bits dentro del registro de instrucciones.
- *Pause-IR*: detiene temporalmente el desplazamiento de la instrucción, y es útil para realizar interrupciones controladas en caso de necesitar un retardo para sincronizar acciones dentro del proceso de carga de la nueva instrucción.
- *Exit2-IR*: estado intermedio en el proceso de escaneo del registro de instrucción que finaliza el desplazamiento de bits dentro del registro de instrucciones.
- *Update-IR*: en este estado, la nueva instrucción capturada y desplazada anteriormente, se carga de forma paralela en el controlador TAP para convertirse en la nueva instrucción activa. Así, se finaliza con el proceso de escaneo del registro de instrucción.

Controlador TMP

Su participación es opcional dentro del proceso de pruebas y su propósito es mantener en modo de prueba el sistema principal, así realiza las verificaciones del dispositivo de forma segura, sin importar que el controlador TAP se encuentre en el estado *test-logic-reset*, aunque por accidente se envíen comandos como *BYPASS* o *PRELOAD*.

Para controlar el funcionamiento del controlador TMP, el estándar IEEE 1149.1 incorpora las instrucciones *CLAM_HOLD, CLAM_RELEASE* y *TMP_STATUS*. Las dos primeras permiten establecer los estados *persistence-off* y *persistence-on*, respectivamente, mientras que *TMP_STATUS* permite consultar el estado del controlador. Este último opera, por tanto, mediante los dos estados antes mencionados.

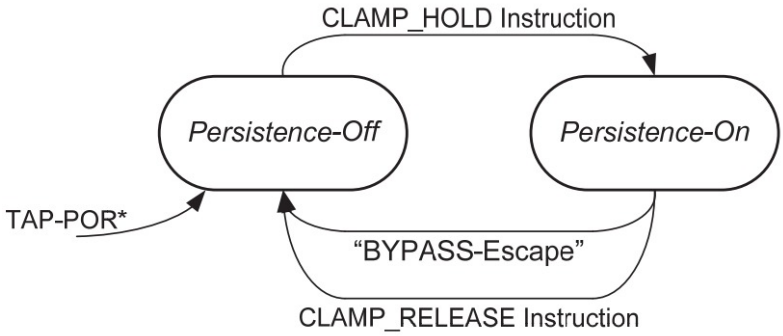


Figura 2.3: Diagrama de estados del controlador TMP.
Fuente: [24].

De acuerdo con el diagrama 2.3, cuando el controlador TMP entra en el estado *persistence-on*, mantiene las pruebas de verificación y diagnóstico. Por el contrario, cuando el controlador TMP ingresa al estado *persitence-off*, el sistema vuelve a su operación normal.

Para que la transición entre estados pueda llevarse a cabo, es necesario que se presente un flanco de subida de TCK junto con la activación de alguna de las instrucciones *CLAMP-HOLD o CLAMP-RELEASE*. Esta transición también puede llevarse a cabo por medio de $\overline{\text{TRST}}$ si se contempla, a través de la señal $\overline{\text{TAP}-\text{POR}}$.

2.2.2.2. Registro de instrucciones

Permite desplazar instrucciones al sistema, ya sea para seleccionar una prueba lógica, un registro de datos, o ambos. Dicho registro debe seguir una serie de reglas para un funcionamiento sincronizado con el resto de componentes:

- El tamaño mínimo que debe tener el registro es de dos bits; ya que al menos debe cumplir con cuatro instrucciones:
  - Seleccionar el registro *BYPASS*.
  - Acceder al registro *boundary-scan* en tres modos diferentes: *EXTEST, PRELOAD* y *SAMPLE*.
- Para evitar fallas en el funcionamiento de la lógica del sistema, la salida del registro de instrucciones debe engancharse; es decir, existe un registro adicional para mantener activa la nueva instrucción en el controlador TAP. Este movimiento se realiza en el flanco de bajada de TCK, cuando el controlador TAP se encuentre en el estado *update-ir*.
- La nueva instrucción se activa en el flanco de bajada de TCK, mientras el controlador TAP se encuentra en el estado *update-ir* o *test-logic-reset*.
- Los dos bits menos significativos del registro de instrucciones deben cargarse con el patrón 01 en el estado *capture-ir* para comprobar el correcto funcionamiento del registro. Si no aparece al leer la salida por TDO, existe un error en la transmisión de datos.
- El desplazamiento de los bits en el registro de instrucciones se realiza en cada flanco de subida de TCK, mientras el controlador TAP se encuentre en el estado *shift-ir*.
- Si el controlador TAP se encuentra en el estado *test-logic-reset*, la instrucción a enganchar debe ser *IDCODE* o *BYPASS*.
- Si se proporciona un nivel bajo en $\overline{\text{TRST}}$, la instrucción a enganchar debe ser *IDCODE* o *BYPASS*. En este caso, el cambio debe realizarse de forma asíncrona.
- Las celdas del registro de instrucciones que se encuentren inactivas, tienen que cargarse con valores fijos para evitar fallas en la lógica del sistema.
- El registro de instrucciones cuenta con dos entradas: una serial, obligatoria en el diseño, que se conecta a la entrada TDI con el registro y obtiene su salida por TDO; y una paralela, opcional, utilizada para cargar instrucciones específicas para realizar pruebas internas. Si la entrada paralela no se emplea, se recomienda cargar valores binarios fijos a esas celdas para evitar incertidumbres en el funcionamiento.

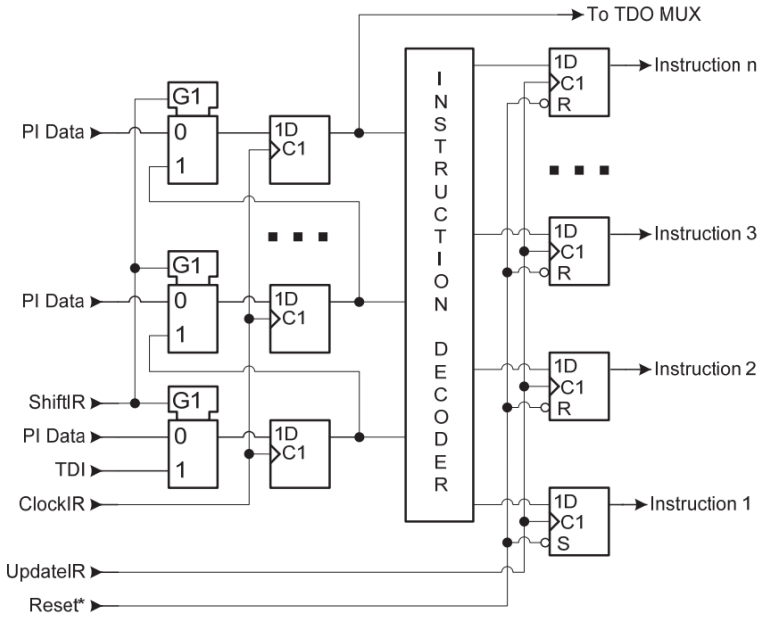


Figura 2.4: Diagrama del registro de instrucciones, que muestra su conexión con el decodificador y las salidas hacia las distintas instrucciones.
Fuente: [24].

2.2.2.3. Instrucciones

Las instrucciones, definidas en la lógica de prueba del estándar, especifican que registros de datos se activan. Cada instrucción debe cumplir ciertas reglas establecidas por el estándar para que realicen sus funciones dentro de la lógica de prueba. A continuación, se mencionan algunas de esas reglas:

- Los registros de datos inactivos deben operar de forma que no interfieran con el funcionamiento de la lógica de prueba.
- Cada instrucción selecciona un camino único serial entre TDI y TDO, a través del cual se desplazan los datos. Sin embargo, pueden seleccionar varios registros de datos conectados en serie.
- Las respuestas lógicas asociadas a la instrucción activa no deben modificarse hasta que se cambie la instrucción durante un nuevo ciclo de escaneo.
- Los códigos binarios de las instrucciones pueden compartirse, siempre que se respeten las reglas del estándar y cada instrucción cumpla con sus funciones de manera independiente.
- Una instrucción se mantiene activa dentro del controlador TAP hasta que una nueva instrucción es cargada en la salida paralela del registro de instrucciones durante los estados *update-ir* o *test-logic-reset*.
- El comportamiento de la lógica de prueba está determinado por el estado del controlador TAP, los datos en los registros de datos y por la instrucción activa.

- El estándar contempla la posibilidad de agregar instrucciones adicionales bajo ciertas consideraciones, por ejemplo:
  - Si el chip del dispositivo JTAG cuenta con un registro de identificación, entonces se agrega la instrucción *IDCODE*.
  - Se debe agregar la instrucción *USERCODE* cuando el dispositivo JTAG sea programable por el usuario, aunque su programación no se realice directamente a través de JTAG.
  - Si el chip cuenta con entradas y salidas (E/S) programables, se deben agregar las instrucciones *INIT_SETUP, INIT_SETUP_CLAMP, INIT_RUN*.
- El decodificador de instrucciones carece de elementos de almacenamiento; por lo tanto, su salida depende únicamente del valor presente en sus entradas y su funcionamiento es meramente combinacional.

A partir de las consideraciones establecidas en el estándar IEEE 1149.1 sobre las instrucciones compatibles con JTAG, se define una clasificación de acuerdo con el nivel de acceso, lo que permite distinguir entre aquellas destinadas al uso del usuario y aquellas reservadas para el fabricante de dispositivos compatibles con dicho estándar.

- Instrucciones accesibles: instrucciones definidas por el estándar disponibles para el usuario. Todo dispositivo compatible con JTAG debe implementar, al menos, las siguientes instrucciones accesibles: *BYPASS, SAMPLE, PRELOAD* y *EXTEST*. La codificación binaria de las instrucciones en el registro de instrucciones es asignada por el fabricante, con la excepción de *BYPASS*, cuya codificación está normada (todos los bits en uno lógico, o en casos permitidos en cero lógico).
- Instrucciones restringidas: instrucciones no disponibles para los usuarios de los dispositivos JTAG, cuya definición y codificación son determinadas por el fabricante. Estas instrucciones complementan a las instrucciones públicas y pueden no estar documentadas en el archivo lenguaje de descripción de escaneo de frontera (BSDL, por sus siglas en inglés).

A continuación, se presenta una explicación breve sobre las instrucciones públicas contempladas en el estándar JTAG.

- Instrucción *BYPASS*: selecciona el registro *BYPASS*, de un solo bit de longitud, conectado entre TDI y TDO durante el estado *shift-dr*. Agiliza el desplazamiento de datos a través de dispositivos que no requieren pruebas internas. Si el componente no posee un registro de identificación, se activa por defecto la instrucción *BYPASS* al ingresar al estado *test-logic-reset*.
- Instrucción *SAMPLE*: permite monitorear el estado del sistema a través de capturas de datos binarios de las señales sin interferir en la lógica de prueba. Si *SAMPLE* produce el mismo efecto que *PRELOAD*, pueden compartir código binario.
- Instrucción *PRELOAD*: carga datos en las salidas paralelas del sistema, y usa los valores almacenados del registro de desplazamiento *boundary-scan*.

- Instrucción *EXTEST*: realiza pruebas y verifica interconexiones físicas entre chips, por medio del envío de datos binarios a través de los registros *boundary-scan*, sin ejecutar lógica interna del chip. El chip se mantiene en estado de espera para evitar procesar información y alterar su lógica interna.
- Instrucción *INTEST*: verifica, a baja velocidad, el funcionamiento de los componentes una vez ensamblados en la tarjeta. El sistema debe operar en modo *single-step*; es decir, por cada carga de estímulo, la lógica realiza una operación, espera al siguiente estímulo y repite el ciclo con la siguiente operación. Al finalizar, se recomienda reiniciar la lógica del sistema para evitar estados indeterminados.
- Instrucción *RUNBIST*: ejecuta pruebas autónomas del componente sin la necesidad de cargar patrones de pruebas complejos ni de operar paso a paso. Su propósito es comprobar el funcionamiento del componente y detectar fallas internas. Los resultados permanecen disponibles tras las pruebas.
- Instrucción *CLAMP*: fija valores constantes en los pines de salida de determinados circuitos integrados, sin que estos valores cambien ni interfieran con las pruebas. El sistema permanece en reposo para evitar alteraciones en su funcionamiento lógico mientras se recibe la instrucción.
- Instrucción *IDCODE*: si el dispositivo cuenta con un registro de identificación, esta instrucción se incluye. Selecciona el registro de identificación que se conecta entre TDI y TDO y se carga el código de identificación del dispositivo.
- Instrucción *USERCODE*: si el dispositivo cuenta con un registro de identificación, se incluye esta instrucción. Permite cargar y desplazar un código binario de 32 bits programable por el usuario.
- Instrucción *ECIDCODE*: selecciona el registro *ECID* conectado entre TDI y TDO. Solicita el valor de identificación electrónica que permite la identificación y ubicación del componente, de lo contrario, muestra solo unos lógicos. La identificación electrónica es un número de serie del chip de silicio. Puede ser usada para verificación de manufactura, rastreo de lotes o producción.
- Instrucción *HIGHZ*: forza todas las salidas del sistema a un estado de alta impedancia. Una vez usada, se debe reiniciar la lógica de prueba para evitar estados indeterminados. La lógica interna del chip se mantiene en un estado de espera.
- Instrucciones *INIT_SETUP e INIT_SETUP_CLAMP*: si se incluye el registro de datos de inicialización y el proceso de inicialización requiere de parámetros adicionales, se deben de incluir. Inicializan entradas y salidas programables con el fin de configurar correctamente la lógica del sistema y dejarla lista para condiciones de pruebas seguras.
- Instrucción *INIT_RUN*: se utiliza en dispositivos cuya secuencia de inicialización requiere de una configuración más compleja que la simple carga de valores estáticos.
- Instrucciones *CLAMP_HOLD, CLAMP_RELEASE y TMP_STATUS*: se deben proporcionar si se incluye el controlador TMP y el registro de estado TMP.

- La instrucción *CLAMP_HOLD* manda al controlador TMP al estado *persistance-on* (figura 2.3), y mantiene las E/S fijas y estables de acuerdo con los datos dentro del registro *boundary-scan*.
- La instrucción *CLAMP_RELEASE* cambia el estado del controlador TMP a *persistance-off* (figura 2.3), y permite que las E/S retomen su funcionamiento normal en la lógica de prueba.
- La instrucción *TMP_STATUS* permite monitorear las condiciones del estado actual del controlador TMP.

- Instrucción *RESET*: selecciona el registro de reset conectado entre TDI y TDO, y se encarga de proporcionar un control de las señales de reinicio y de otras señales relacionadas con la lógica del sistema mediante el controlador TAP.

2.2.2.4. Registro de datos

Los registros de datos son otros elementos fundamentales dentro de la arquitectura JTAG. A través de ellos se permite el desplazamiento de datos tanto para reconocimiento, realización de pruebas de validación o verificación y para configurar los dispositivos compatibles con el estándar. Estos registros son conectados entre TDI y TDO y trabajan en conjunto con el controlador TAP, el registro de instrucciones y las señales del puerto TAP (figura 2.5).

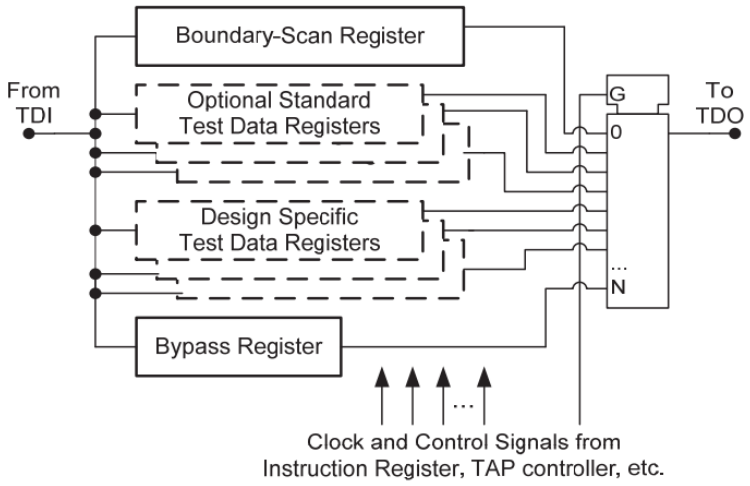


Figura 2.5: Estructura general de los registros de datos definidos por el estándar IEEE 1149.1.
Fuente: [24].

El estándar establece que todos los dispositivos compatibles con JTAG deben contar con dos registros de datos fundamentales.

- Registro *BYPASS*: registro serial con longitud mínima de una celda. Permite el paso de datos a través de un componente sin afectar su funcionamiento interno. Si el dispositivo no cuenta con el registro de identificación, se le carga un uno lógico a este registro para reflejarse en la salida, lo que lo vuelve un método para identificar cuales dispositivos cuentan con un registro de identificación.
- Registro *boundary-scan*: registro serial que permite realizar pruebas de interconexión de tarjeta. Identifica defectos en la fabricación como cortocircuitos o conexiones abiertas. También otorga acceso a las entradas y salidas de los componentes al probar su lógica interna. Su complejidad se basa en como está conectado alrededor del chip, su comportamiento ante ciertas instrucciones y por su adaptabilidad a diferentes tipos de señales.

Además, el estándar permite agregar otros registros de prueba cuando sean necesarios. Esto depende de las necesidades de la lógica de prueba o del sistema. Entre estos registros opcionales se encuentran:

- Registro de identificación del dispositivo: se trata de un registro de desplazamiento que permite que la identificación del dispositivo sea obtenida por medio del controlador TAP. Con esta información, se identifica el archivo BSDL y los patrones de prueba correctos para la tarjeta.
- Registro de identificación electrónica del chip: complementa la identificación del dispositivo y los códigos de usuario. Permite obtener un identificador único para distinguir una instancia específica del componente. De esta forma, se pueden conservar datos históricos sobre el componente como su fabricación, prueba, uso, etc.
- Registro de datos de inicialización: proporciona parámetros para inicializar puertos de E/S y otros circuitos que deben configurarse antes de ejecutar pruebas de interconexión con el registro *boundary-scan*.
- Registro de estado de inicialización: se trata de un registro de solo lectura. Permite conocer el estado de un procedimiento de inicialización mediante el controlador TAP y el fabricante tiene libertad para definir qué estados se capturan.
- Registro de estado TMP: solo se incluye en el diseño en caso de que se proporcione el controlador TMP. Su propósito es dar a conocer el estado actual del controlador TMP.
- Registro de selección de reinicio: permite controlar las funciones de reinicio del sistema; además, permite bloquear reinicios no deseados a la lógica del sistema durante las pruebas.

Si se incluyen registros adicionales, sus respectivas instrucciones también deben estar presentes. Estas pueden aparecer como restringidas o accesibles. Asimismo, todo registro de

datos de prueba debe cumplir con ciertas características que eviten interferencias con el funcionamiento de la lógica de prueba y comportamientos no deseados durante su operación. A continuación, se presentan algunas de esas características que deben considerarse en este tipo de registros:

- Los datos introducidos por TDI deben aparecer invertidos por TDO después de cierta cantidad de ciclos de reloj. El número de ciclos de reloj está determinado por la longitud del registro que deben recorrer los datos.
- La longitud de los registros debe ser fija e independiente de la instrucción con la que se accede a ese registro.
- Los registros deben tener una longitud mínima de uno.
- Cada registro debe ser capaz de responder al controlador TAP.
- Los registros pueden usarse para ejecutar funciones de prueba, a menos que esté prohibido.
- Cuando el controlador TAP se encuentre en el estado *shift-dr*, los registros deben desplazar datos en cada flanco de subida de TCK.
- Cuando se requiera que un registro cargue datos desde una entrada paralela en respuesta a la instrucción activa, debe hacerlo en el flanco de subida de TCK cuando el controlador TAP se encuentre en el estado *capture-dr*.
- Si la instrucción *RUNBITS* se encuentra activa, la respuesta del registro debe realizarse en el estado *run-test/idle*.
- Los registros de datos que no estén en funcionamiento, deben de funcionar de tal forma que no interfieran con la lógica del sistema del chip.
- A pesar de que existen instrucciones que pueden seleccionar más de un registro para realizar su tarea, solo un registro debe estar conectado entre TDI y TDO.

Existe otro componente crucial dentro del funcionamiento de los registros de datos y son los segmentos. Básicamente se trata de celdas que se pueden concatenar con otros registros, o incluso con otros segmentos, para crear nuevas rutas que permiten seleccionar porciones específicas de diseño para incluir o excluir ciertas funciones según sea necesario. Los segmentos se clasifican en dos categorías, de acuerdo con el estándar, las cuales son:

- Excluibles: siempre presentes en el diseño, pero pueden ser omitidos al escanear.
- Seleccionables: son activados o desactivados de acuerdo con la codificación de la operación lógica a realizar.

Estos segmentos se activan o no por medio de celdas de selección de segmentos, controlan si un segmento excluible o seleccionable debe incluirse y, al igual que los registros de datos, estos segmentos están sujetos a reglas definidas por el estándar para realizar pruebas lógicas en función de la instrucción activa.

2.3. Interfaz periférica serial (SPI)

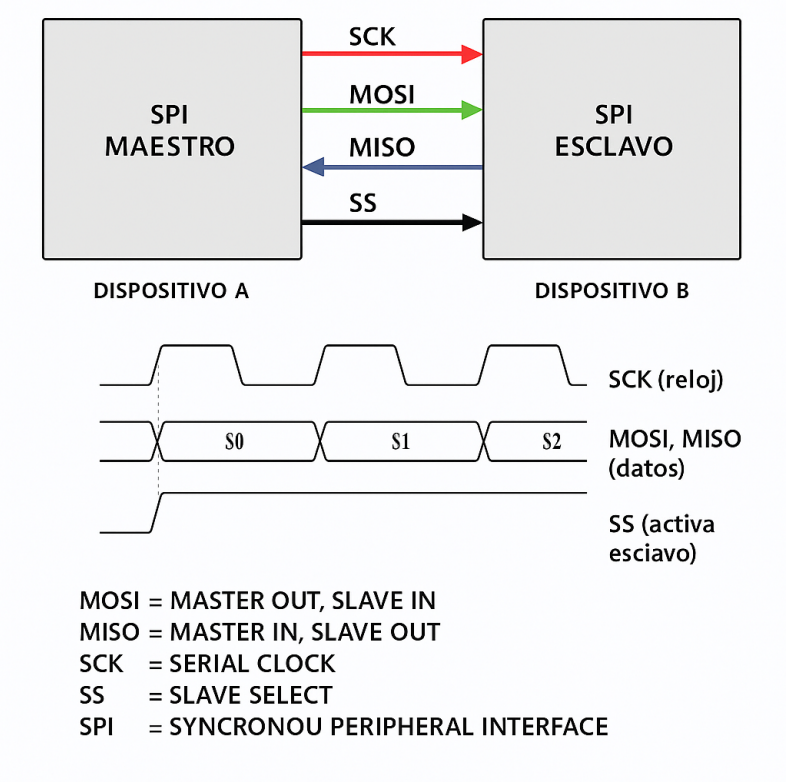


Figura 2.6: Estructura general de SPI.
Fuente: [26].

Con base en [27, 28, 26] SPI es un protocolo de comunicación serial síncrona desarrollado por *Motorola* a mediados de la década de 1980. Diseñada para proporcionar una transferencia de datos entre un dispositivo maestro y uno o más dispositivos esclavos, como se muestra en la figura 2.6.

Esta interfaz define las líneas físicas de comunicación y sincronización, así como las reglas del protocolo para la transferencia y el control de información binaria. SPI ofrece una comunicación serial síncrona *full-duplex*; es decir, la transmisión y recepción de información binaria es simultánea bajo el control de un reloj controlado por el dispositivo maestro. Empleado ampliamente en comunicación de corto alcance con dispositivos como memorias, convertidores de señales, pantallas gráficas y sensores, entre otros chips en sistemas integrados. Es gracias a estas características que SPI ha adquirido gran importancia en la comunicación digital moderna.

2.3.1. Arquitectura y señales

Existen dos actores principales quienes construyen la comunicación, los cuales son:

- Dispositivo maestro: controla las operaciones de comunicación; es decir, gestiona las líneas de transmisión y genera la señal de reloj con la cual se da la indicación de transmitir y recibir un bit de información por cada pulso de reloj. Es el encargado de controlar rigurosamente el flujo de datos en toda la red SPI.
- Dispositivo esclavo: se trata de todo periférico conectado a la red SPI. Cada uno de los dispositivos esclavos es controlado por el dispositivo maestro dentro de la red, el cual, a través de una línea de selección decide con cual periférico comunicarse, lo que le permite el intercambio de información binaria para un rendimiento eficiente del sistema.

SPI cuenta con cuatro señales y son la base de la comunicación entre el dispositivo maestro y el dispositivo esclavo. A continuación, se mencionan:

- Reloj serial (SCLK, por sus siglas en inglés): es el reloj de la comunicación síncrona de SPI. Se encuentra activa mientras exista comunicación en la red y es controlado por el dispositivo maestro. Por cada pulso de reloj, un bit de información ingresa al dispositivo maestro y al esclavo, por lo tanto, se requieren de 8 pulsos de reloj para transferir un byte completo.
- Salida del maestro - entrada del esclavo (MOSI, por sus siglas en inglés): señal portadora de la información binaria enviada por el dispositivo maestro al dispositivo esclavo seleccionado.
- Entrada del maestro - salida del esclavo (MISO, por sus siglas en inglés): señal portadora de la respuesta binaria del dispositivo esclavo hacia el dispositivo maestro.
- Selección del chip/selección del esclavo (CS/SS, por su siglas en ingles): señal de selección, con ella se puede habilitar al esclavo con el que se desea realizar la comunicación.

En cuanto a sus diferentes configuraciones en la red, el protocolo SPI está basado en un esquema maestro - esclavo. Como se mencionó, un único maestro gestiona la comunicación dentro de la red SPI mediante SCLK y CS/SS para habilitar al esclavo con el que se desea entablar comunicación. Este enfoque permite construir diferentes configuraciones en la red, desde la comunicación directa entre un único maestro y esclavo, hasta la comunicación con múltiples esclavos bajo el control de un mismo maestro. A continuación se detalla cada uno de ellas:

- Maestro y esclavo único: corresponde a la configuración básica de una red SPI, constituida por un dispositivo maestro y un único dispositivo esclavo. El maestro controla la comunicación y genera la señal de reloj necesaria para realizar el intercambio de datos.
- Multiesclavo: en esta configuración el maestro está conectado a múltiples dispositivos esclavos dentro de la red SPI, por lo tanto, es importante tener una señal CS/SS para cada esclavo. Dada la anterior característica, conforme existan más esclavos en la red, más señales CS/SS para cada uno de ellos son necesarias, por lo que se considera una gran limitante en cuanto al número de periféricos conectados en la red.
- *Daisy - chain*: como su nombre lo indica, se trata de una configuración en cadena donde la señal CS/SS también está compartida con cada uno de los dispositivos esclavos dentro de la red SPI, y la información binaria se propaga de un dispositivo esclavo al siguiente en la cadena. Todos los dispositivos en la red reciben el mismo reloj al mismo tiempo. Sin embargo, uno de los problemas que tiene esta configuración es que se requiere un número mayor de pulsos de reloj para llegar a dispositivos esclavos cada vez más lejanos del dispositivo maestro.

2.3.2. Modos y parámetros de operación

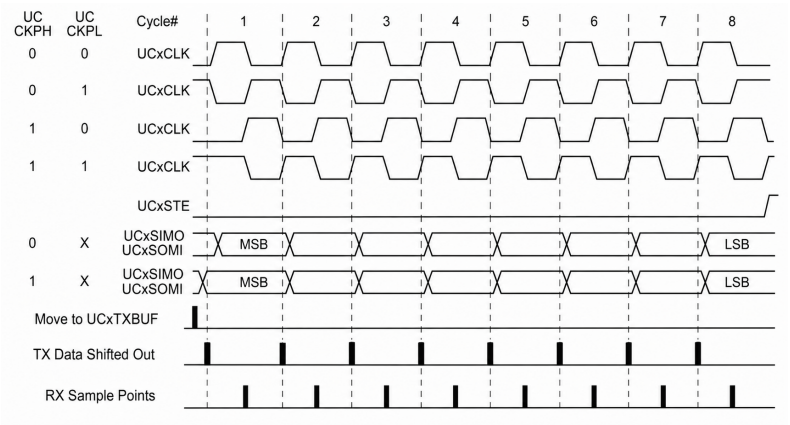


Figura 2.7: Modos de operación de SPI.
Fuente: [29].

SPI da la oportunidad de modificar parámetros para establecer la forma en como se realiza la comunicación entre el dispositivo maestro y el dispositivo esclavo, como se muestra en la figura 2.7, lo que permite garantizar compatibilidad y fiabilidad en la transmisión. Entre los parámetros más importantes se encuentran la polaridad y fase de SCLK, la frecuencia de operación, el orden y tamaño de palabra. Estas configuraciones son las que hacen a SPI un protocolo de comunicación versátil y compatible con un gran número de periféricos [27, 29].

Existen cuatro modos de configuración para SCLK. Estos modos son posibles gracias a las diferentes configuraciones de la polaridad del reloj (CPOL, por sus siglas en inglés) y de la fase del reloj (CPHA, por sus siglas en inglés):

- CPOL: es el estado en el que se debe encontrar SCLK una vez que ya no existe comunicación o que el bus se encuentra en estado inactivo. Puede ser en estado bajo o en estado alto.
- CPHA: parámetro que determina en que flanco de SCLK se realiza el muestreo de la información y en cual se efectúa la actualización de los datos en las líneas de comunicación. De acuerdo con la posibilidad de configuración, los datos pueden ser muestreados en el flanco ascendente o en el flanco descendente.

La figura 2.7 muestra de forma gráfica los cuatro modos de operación descritos anteriormente en el protocolo SPI.

El orden y tamaño de la palabra influyen directamente tanto en la longitud de la trama como en el bit que se envía primero durante la comunicación. Estos aspectos tienen que considerarse puesto que algunos dispositivos esclavos operan con el bit más significativo (MSB, por sus siglas en inglés) o con el bit menos significativo (LSB, por sus siglas en inglés) como el primer bit transmitido; además de que pueden soportar tamaños de palabras diferentes de 8 bits.

Otro parámetro relevante en la operación de SPI es la frecuencia de SCLK, ya que esta determina la velocidad con la que se realiza el intercambio de datos. A diferencia de otros protocolos que establecen modos de operación con frecuencias definidas. SPI no especifica una frecuencia máxima única, pues esta depende de las características temporales y eléctricas de los dispositivos que integran la comunicación. Conforme aumenta la frecuencia, los retardos de propagación, los tiempos de establecimiento y retención, así

como los efectos asociados con la capacitancia y la integridad de las señales, reducen el margen disponible para realizar correctamente la transferencia. Por ello, la frecuencia máxima de operación está determinada por las especificaciones del dispositivo maestro, de los esclavos y de la interconexión entre ellos [30].

2.3.3. Ventajas y limitaciones

Con base en [31, 32], SPI presenta una serie de ventajas y limitaciones que deben considerarse al seleccionar este protocolo para una aplicación determinada. A continuación se dan a conocer las múltiples ventajas que puede proporcionar al momento de entablar comunicación dentro de una red con diferentes dispositivos esclavos:

- SPI permite alcanzar velocidades de transferencia elevadas en comparación con otros protocolos de comunicación serial, como I2C o UART. La frecuencia máxima de operación no está definida de manera única por el protocolo, sino que depende de las características temporales y eléctricas de los dispositivos interconectados. Esto permite adaptar la velocidad de comunicación a las capacidades de los dispositivos empleados.
- Las líneas de SPI utilizan típicamente salidas de tipo *push-pull* para establecer los niveles lógicos durante la transferencia, a diferencia de interfaces de colector o drenador abierto como I2C. Esto simplifica la interconexión y evita que los tiempos de subida de las señales dependan de la combinación entre las resistencias *pull-up* y la capacitancia del bus.
- Comunicación *full-duplex;* es decir, es capaz tanto de transmitir como de recibir información al mismo tiempo a través de sus líneas de transmisión MOSI y MISO.
- La selección del dispositivo esclavo se realiza mediante una línea CS/SS dedicada, por lo que no es necesario transmitir una dirección del dispositivo como parte de la comunicación. Esto reduce la información de control requerida antes del intercambio de datos.
- SPI presenta una reducida sobrecarga de protocolo, ya que la transferencia no incorpora mecanismos de direccionamiento ni confirmación de recepción propios de otras interfaces. En consecuencia, una mayor proporción de los ciclos de comunicación puede destinarse directamente a la transferencia de información.

Sin embargo, como en un principio se mencionó, SPI fue la respuesta a una comunicación sencilla y de corto alcance, por lo tanto, también cuenta con algunas desventajas las cuales son:

- SPI está orientado principalmente a comunicaciones de corta distancia. Conforme aumenta la longitud de las interconexiones, los efectos asociados con la integridad de las señales y los retardos de propagación pueden limitar la frecuencia de operación y comprometer la correcta transferencia de los datos.

- En una configuración con múltiples esclavos, cada dispositivo requiere de su propia señal CS/SS. Por lo tanto, el número de pines y conexiones requeridas aumenta conforme se incorporan más esclavos a la red, lo que puede limitar su escalabilidad frente a interfaces que emplean direccionamiento sobre líneas compartidas, como I2C.
- SPI no incorpora de manera inherente mecanismos de confirmación de recepción o verificación de errores que permitan determinar si la transferencia de información se realizó correctamente. Por ello, cuando la aplicación requiere de este tipo de verificación, es necesario implementarla mediante mecanismo adicionales a niveles de *software* o del protocolo empleado sobre SPI.
- SPI no define de manera inherente mecanismos de arbitraje para configuraciones con múltiples maestros. Por esta razón, una implementación multimaestro requiere lógica o mecanismos adicionales que permitan determinar qué dispositivo obtiene el control del bus y evitar conflictos entre transmisiones. Esta necesidad incrementa los requerimientos de implementación respecto a una configuración con un único maestro.

2.4. Tecnologías de almacenamiento

Las tecnologías de almacenamiento constituyen un elemento fundamental en los sistemas embebidos, ya que permiten la conservación y gestión de información necesaria para el procesamiento, configuración y operación de dispositivos electrónicos. Estas pueden clasificarse en volátiles y no volátiles. Las primeras requieren alimentación eléctrica para conservar la información almacenada, mientras que las segundas mantienen los datos aun en ausencia de energía. Las diferencias entre las tecnologías de memorias no se limitan a la persistencia de la información, sino que también abarcan aspectos como velocidad de acceso, capacidad de almacenamiento, consumo de potencia, ciclos de borrado/escritura y mecanismos físicos de almacenamiento. En esta sección se analizan los fundamentos físicos y operativos de algunas tecnologías de almacenamiento empleadas en sistemas embebidos con aplicaciones espaciales, así como sus características de desempeño más relevantes, con el objetivo de establecer un panorama comparativo que permita comprender sus ventajas, limitaciones y criterios de selección.

2.4.1. Clasificación

De acuerdo con [33], una forma de clasificar a las memorias es dentro de dos categorías: memorias volátiles y no volátiles.

Memorias volátiles

Son dispositivos de almacenamiento que requieren un suministro continuo de energía para conservar la información almacenada; al interrumpir la alimentación, los datos se pierden. Esta propiedad constituye su principal diferencia respecto a las memorias no volátiles. Algunas dentro de esta categoria se utilizan principalmente para almacenar

temporalmente información que debe estar disponible durante la operación del sistema y, en términos generales, presentan menores tiempos de acceso que diversas tecnologías de memoria no volátil [34].

Memorias no volátiles

Son dispositivos de almacenamiento capaces de conservar la información aun cuando se interrumpe el suministro de energía. Esta característica permite utilizarlas para almacenar datos que deben persistir entre ciclos de encendido y apagado del sistema. Dependiendo de la tecnología empleada, presentan diferencias en parámetros como tiempo de acceso, capacidad de almacenamiento, número a ciclos de escritura y retención de datos [35].

Aunque las memorias volátiles y no volátiles presentan diferentes mecanismos de almacenamiento y características de operación, ambas cumplen funciones complementarias dentro de los sistemas embebidos. Las primeras se emplean principalmente para almacenar información durante la ejecución del sistema, mientras que las segundas permiten conservar datos, instrucciones de arranque, configuraciones o programas aun cuando se interrumpe la alimentación. Por esta razón, es común que los sistemas embebidos integren ambos tipos de memoria de acuerdo con los requerimientos de almacenamiento y operación de cada aplicación.

2.4.2. Fundamentos físicos de las tecnologías de almacenamiento

Las tecnologías de memoria empleadas en la industria se distinguen principalmente por el mecanismo físico que emplean para el almacenamiento de información, por ejemplo: cargas eléctricas, fenómenos magnéticos o polarización ferroeléctrica. Es a través de estos mecanismos que se determinan características técnicas como volatilidad, potencia, velocidad de acceso o tolerancia a la radiación. A continuación, se analizan algunas de las tecnologías de memoria utilizadas en sistemas embebidos, con énfasis en sus fundamentos físicos de operación.

- Memoria estática de acceso aleatorio (SRAM, por sus siglas en inglés) [36, 37]: memoria volátil que almacena la información por medio de celdas biestables constituidas, generalmente, por seis transistores de efecto de campo metálico-óxido-semiconductor (MOSFET, por sus siglas en inglés). Cuatro de estos transistores forman dos inversores semiconductores complementarios de metal-óxido (CMOS, por sus siglas en inglés) realimentados de manera cruzada, constituyendo un biestable (*latch*), mientras que los dos transistores restantes permiten el acceso a la celda durante operaciones de lectura y escritura, como se muestra en la figura 2.8. El almacenamiento se logra gracias a dos nodos complementarios ($\mathrm{Q}\, y\, \overline{\mathrm{Q}}$):

  - Cuando $\mathrm{Q} = 1$ y $\overline{\mathrm{Q}} = 0$, la celda guarda 1.
  - Cuando $\mathrm{Q} = 0$ y $\overline{\mathrm{Q}} = 1$, la celda guarda 0.

  La escritura ocurre al activar la línea de palabra (WL, por sus siglas en inglés) y aplicar el valor lógico en las líneas de bit (BL, $\overline{\mathrm{BL}}$, por sus siglas en inglés), con un voltaje igual al de la alimentación del circuito, típicamente entre $1 - 3\,\mathrm{V}$, según las

especificaciones eléctricas del dispositivo. La lectura ocurre al activar WL y detectar el estado almacenado en BL y $\overline{\text{BL}}$. Debido a su alta velocidad de acceso, se emplea en memorias *caché* y *buffers* en sistemas digitales.

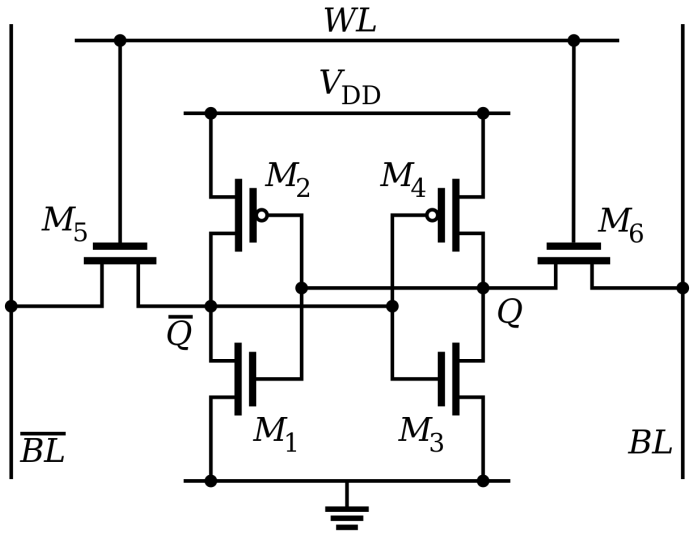


Figura 2.8: Celda de memoria SRAM.
Fuente: [37].

- Memoria dinámica de acceso aleatorio (DRAM, por sus siglas en inglés) [36, 37]: memoria volátil que almacena cada bit como carga eléctrica en un capacitor controlado por un transistor tipo MOSFET, en arquitectura de un transistor - un capacitor (1T-1C), como se muestra en la figura 2.9. Cada celda guarda un bit según el estado del capacitor:

  - Capacitor cargado $\rightarrow$ 1.
  - Capacitor descargado $\rightarrow$ 0.

  La escritura ocurre al activar WL y aplicar el valor lógico en BL, lo que carga o descarga el capacitor con un voltaje cercano al de alimentación del circuito ($1-3{,}3\,\text{V}$), según las especificaciones eléctricas del dispositivo. La lectura ocurre al activar WL y leer la carga del capacitor en BL. Debido a la fuga de carga, esta memoria requiere de ciclos periódicos de refresco. Se emplea como memoria principal en computadoras, servidores, dispositivos móviles, entre otros sistemas.

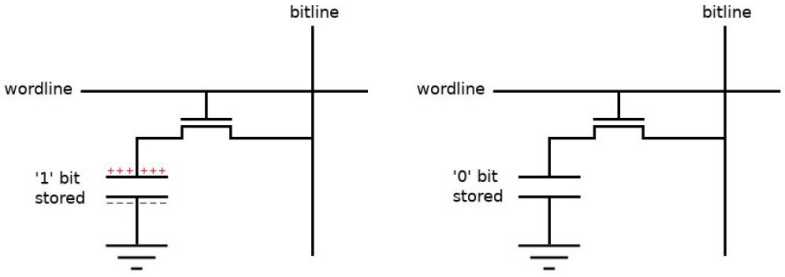


Figura 2.9: Celda de memoria DRAM.
Fuente: [37].

- Memoria de solo lectura programable y borrable (EPROM, por sus siglas en inglés) [38, 39]: memoria no volátil que se basa en transistores tipo MOSFET con puerta flotante, como se muestra en la figura 2.10. El valor lógico depende de la carga almacenada en la puerta flotante:

  - Con carga almacenada $\rightarrow 0$.
  - Sin carga almacenada $\rightarrow 1$.

  La escritura ocurre al aplicar un alto voltaje en la compuerta de control, en un rango entre $12 - 25\,\text{V}$, lo que inyecta electrones en la puerta flotante. La lectura ocurre al aplicar un voltaje de operación y medir la conducción del transistor. A diferencia de otras memorias, la operación de borrado se realiza cuando se expone el circuito a la luz ultravioleta para eliminar la carga almacenada. Debido a su capacidad de conservar datos durante largos periodos, se utiliza para almacenar código de programa, configuraciones y datos de *software*.

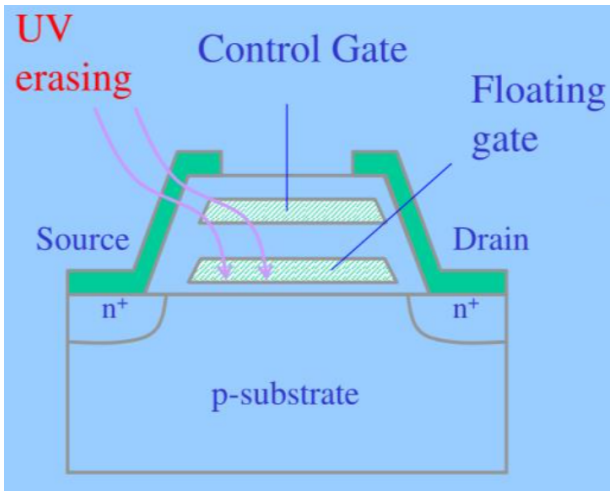


Figura 2.10: Celda de memoria EPROM.
Fuente: [40].

- Memoria de solo lectura programable y borrable eléctricamente (EEPROM, por sus siglas en inglés) [38, 39]: memoria no volátil basada en transistores tipo MOSFET con puerta flotante, como se muestra en la figura 2.11. Su principio de almacenamiento es similar al de la EPROM; sin embargo, permite realizar las operaciones de programación y borrado mediante señales eléctricas, sin necesidad de luz ultravioleta. El dato almacenado depende de la carga eléctrica presente en la puerta flotante:

  - Con carga almacenada $\rightarrow 0$.
  - Sin carga almacenada $\rightarrow 1$.

  La escritura y borrado se realizan gracias al efecto túnel de *Fowler-Nordheim*, al aplicar un voltaje elevado entre $10 - 20\,\text{V}$, en la compuerta de control. La lectura ocurre al aplicar el voltaje de operación y medir la conducción del transistor. Debido a su capacidad de reprogramación eléctrica, se utiliza para almacenar configuraciones y pequeños volúmenes de datos en sistemas electrónicos.

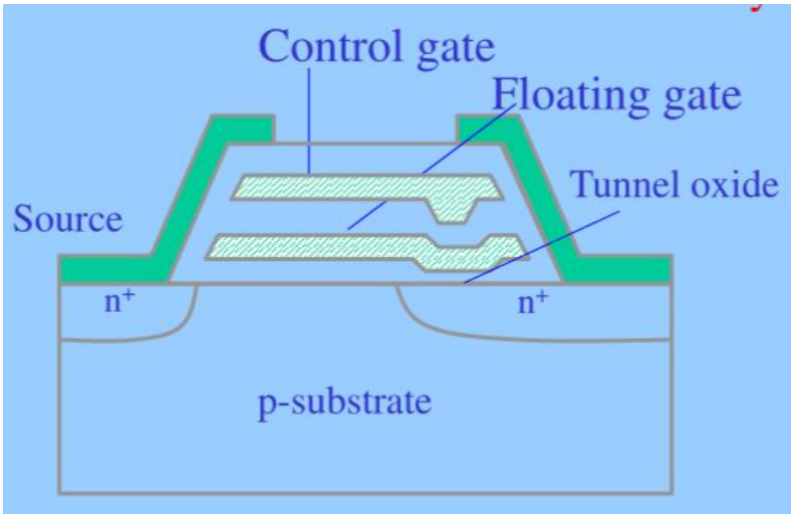


Figura 2.11: Celda de memoria EEPROM.
Fuente: [40].

- Memoria *flash* [41]: memoria no volátil basada en transistores tipo MOSFET con puerta flotante, como se muestra en la figura 2.12. El valor almacenado depende de la carga en la puerta flotante:

  - Puerta flotante con electrones atrapados $\rightarrow 0$.
  - Puerta glotante sin electrones atrapados $\rightarrow 1$.

  La escritura ocurre al inyectar electrones en la puerta flotante en un rango alto de voltaje, entre $10 - 20\,\text{V}$. La lectura ocurre al aplicar el voltaje de operación y medir la conducción del transistor.

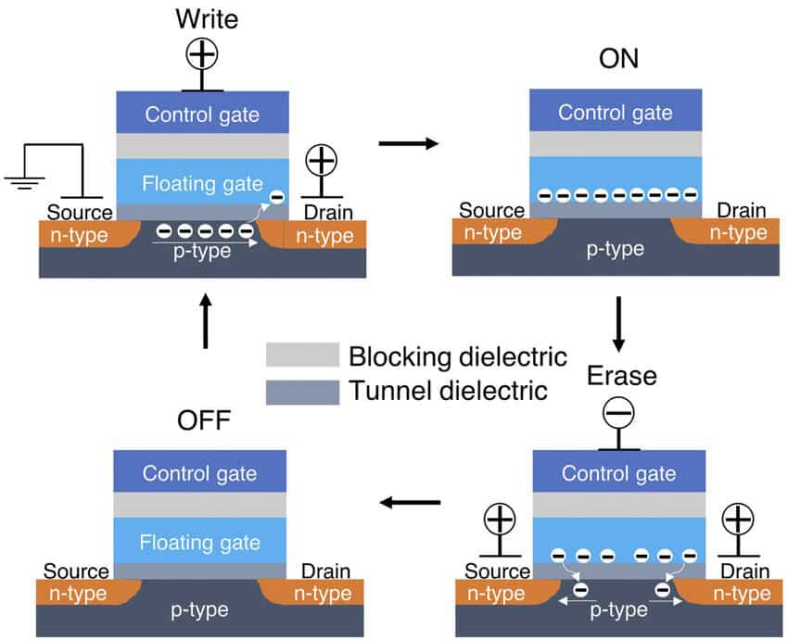
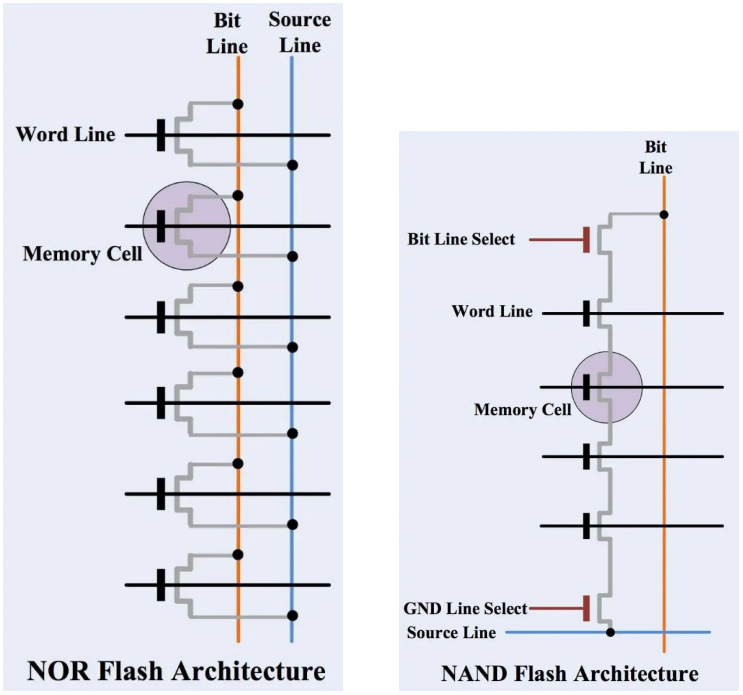


Figura 2.12: Celda de memoria *flash*.
Fuente: [41].

Dentro de este tipo de memorias destacan dos arquitecturas principales: NOR *flash* y NAND *flash*:

- Memoria NOR *flash* [41, 42]: Sus celdas se conectan en paralelo con BL, lo que proporciona acceso directo y aleatorio a cada celda, como se muestra en la figura 2.13a. La escritura ocurre al inyectar electrones a la puerta flotante por medio de altos niveles de voltaje, entre $10 - 20\,\text{V}$. El borrado ocurre gracias al efecto túnel de *Fowler-Nordheim* que extrae los electrones almacenados. La lectura ocurre al aplicar el voltaje de operación y medir la conducción del transistor. Debido a su acceso aleatorio, se utiliza principalmente para almacenar *software* en sistemas embebidos.
- Memoria NAND *flash* [41, 43]: Sus celdas se conectan en serie, como se muestra en la figura 2.13b, lo que reduce el área y aumenta la densidad de almacenamiento. La escritura ocurre mediante inyección de electrones en la puerta flotante con voltajes típicos entre $10 - 20\,\text{V}$, mientras que el borrado utiliza el efecto de túnel de *Fowler-Nordheim* para extraer electrones almacenados. La lectura ocurre al aplicar el voltaje de operación y detectar la conducción del arreglo de celdas. Debido a su alta densidad y menor costo por bit, se emplea en dispositivos de almacenamiento masivo como memorias microSD y unidades de estado sólido (SSD, por sus siglas en inglés), entre otros.



(a) Arquitectura de la memoria NOR *flash*.

(b) Arquitectura de la memoria NAND *flash*.

Figura 2.13: Arquitecturas de la memoria *flash*. Fuente: [41].

- Memoria ferroeléctrica de acceso aleatorio (FRAM, por sus siglas en inglés) [44, 45]: memoria no volátil que almacena datos mediante la polarización de un material ferroeléctrico. Una celda convencional está formada por un transistor tipo MOSFET y un capacitor ferroeléctrico, como se muestra en la figura 2.14. La información se representa por dos estados de polarización del material ferroeléctrico:

  - Una orientación de polarización → 1.
  - La orientación opuesta → 0.

  La escritura ocurre al aplicar un campo eléctrico que establece una de estas orientaciones, mientras que la lectura se basa en la respuesta eléctrica producida al modificar o detectar el estado de polarización.

  Una característica relevante de esta tecnología es su tolerancia a los efectos de radiación. A diferencia de memorias cuyo estado lógico depende del almacenamiento de carga eléctrica, la FRAM conserva la información mediante la polarización remanente del material ferroeléctrico. Por esta razón, la celda de almacenamiento puede presentar una menor susceptibilidad a la degradación asociados con la acumulación o pérdida de carga inducida por radiación. Esta característica hace de la FRAM una alternativa de interés para aplicaciones aeroespaciales y otros sistemas sometidos a ambientes de radiación.

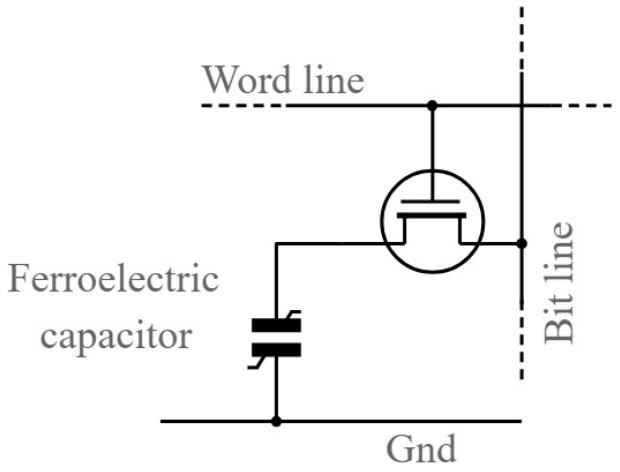


Figura 2.14: Celda de memoria FRAM.
Fuente: [44].

- Memoria magnetoresistiva de acceso aleatorio (MRAM, por sus siglas en inglés) [46, 47]: memoria no volátil que almacena datos en una unión de túnel magnético, formada por dos capas ferromagnéticas separadas por una capa dieléctrica fina, como se muestra en la figura 2.15. Una capa mantiene magnetización fija y la otra puede cambiar su orientación. El dato se determina por la orientación relativa de ambas capas:

  - Orientación paralela → menor resistencia → 0.
  - Orientación antiparalela → mayor resistencia → 1.

La escritura ocurre mediante la aplicación de pulsos de corriente eléctrica en un rango común entre $1 - 3\,\mathrm{V}$, según las especificaciones eléctricas del dispositivo, que cambian la orientación magnética de la capa libre. La lectura se realiza a través de la variación de resistencia producida por el efecto túnel. Una característica relevante de la MRAM para aplicaciones espaciales es la tolerancia a la radiación. A diferencia de las memorias cuyo estado lógico depende del almacenamiento de carga eléctrica, en la MRAM la información se representa mediante la orientación relativa de la magnetización de sus capas ferromagnéticas. Por esta razón, el elemento de almacenamiento presenta menor susceptibilidad a mecanismos asociados con la generación, acumulación o pérdida de carga inducida por radiación.

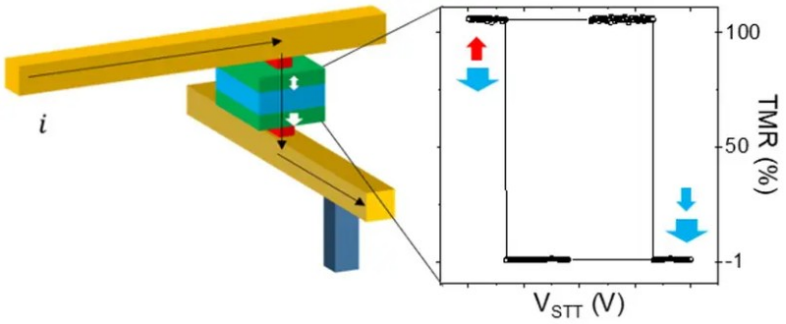


Figura 2.15: Celda de memoria MRAM.
Fuente: [47].

2.4.3. Comparación tecnológica

Una vez presentados los mecanismos físicos de cada tecnología de almacenamiento, se comparan sus principales parámetros para determinar las más adecuadas en aplicaciones de sistemas embebidos del sector espacial.

Memoria / Parámetros	SRAM	DRAM	EPROM	EEPROM	NOR *flash*	NAND *flash*	FRAM	MRAM
Volatilidad	Sí	Sí	No	No	No	No	No	No
Tamaño de celda (F^2)	≈ 120	$6-8$	$20-40$	$20-40$	$10-12$	$4-6$	$20-30$	$20-40$
Ciclos de borrado/escritura	Alta $\approx 10^{15}$	Alta $\approx 10^{15}$	Baja $\approx 10^3$	Baja 10^5-10^6	Baja $\approx 10^5$	Baja $\approx 10^5$	Alta $\approx 10^{12}$	Alta $\approx 10^{15}$
Tiempos de lectura	$2-10$ ns	$50-100$ ns	En el orden de ms	En el orden de ms	<80 ns	$\approx 20\,\mu$s	$20-80$ ns	$3-20$ ns
Tiempo de escritura	$2-10$ ns	$50-100$ ns	En el orden de ms	En el orden de ms	$<25\,\mu$s por palabra	$\approx 0{,}5$ ms por sector	50 ns	$3-20$ ns
Tiempo de borrado	$2-10$ ns	$50-100$ ns	Chip completo $15-30$ min	En el orden de ms	≈ 1 s por sector	$\approx 0{,}5$ ms por sector	$3-20$ ns	≈ 50 ns
Voltaje para escribir (V)	$1-3$	$1-3{,}3$	$12-25$	$10-20$	$10-20$	$10-20$	$1{,}5-3{,}3$	$1-3$
Precio por bit (USD)	$\approx 0{,}01$	$\approx 0{,}001$	$\approx 0{,}01$	$\approx 0{,}01$	$\approx 0{,}001$	$\approx 0{,}0001$	$\approx 0{,}001$	$\approx 0{,}001$
Retención de datos (años)	No retiene datos	No retiene datos	$10-20$	$10-20$	$10-20$	$5-20$	≈ 10	>10
Tolerancia a la radiación TID (krad)	190 [48]	$20-60$ [49]	124 [50]	90 [50]	$\approx 14{,}2$ [51]	<20 [52]	En el orden de Mrad [53]	$1000-10000$ [52]
LETh ($\text{MeV}\cdot\text{cm}^2/\text{mg}$)	$1{,}7-2{,}4$ [54]	0,4776 [55]	1,4 [56]	$10-20$ [57]	≈ 10 [51]	<2 [52]	Inmune (celdas) [53]	69 [52]

Tabla 2.1: Comparación de características tecnológicas de desempeño y tolerancia a la radiación de las memorias SRAM, DRAM, EPROM, EEPROM, NOR *flash*, NAND *flash*, MRAM y FRAM. Los valores presentados son aproximados y pueden variar en función del fabricante, tecnología de fabricación, lote y condiciones de caracterización.
Elaboración propia con base en: [36, 58, 59, 60, 61, 62, 52, 48, 54, 49, 55, 50, 56, 57, 51, 53].

De acuerdo con la tabla 2.1, las memorias analizadas presentan diferencias en volatilidad, velocidad de acceso, números de ciclos de borrado/escritura, retención de datos y tolerancia a la radiación. Por ello, su selección depende de las especificaciones técnicas de cada proyecto. En este trabajo, los parámetros de mayor interés son TID y LETh, ya que permiten evaluar la susceptibilidad de cada tecnología ante la acumulación de radiación y los efectos inducidos por partículas energéticas.

Bajo este criterio, las memorias MRAM y FRAM destacan como las opciones más atractivas por sus tecnologías de almacenamiento (subsección 2.4.1), y a la tolerancia a la radiación que pueden presentar. Esta característica resulta especialmente relevante para aplicaciones en ambientes de radiación [63, 64]. De manera adicional, ambas tecnologías presentan características favorables en términos de tiempo de acceso y número de ciclos de borrado/escritura, aspectos que pueden resultar ventajoso dependiendo de las especificaciones técnicas de la aplicación. Sin embargo, un estudio llevado a cabo por la administración nacional de aeronáutica y del espacio (NASA, por sus siglas en inglés) señala que, aunque la celda de memoria puede ser tolerante a la radiación, los circuitos periféricos basados comúnmente en CMOS, cuentan con una tolerancia limitada [65]. Por esta razón, su uso debe evaluarse en función del comportamiento del sistema completo.

Capítulo 3

Estado del arte

En proyectos con aplicaciones espaciales, especialmente en plataformas de bajo costo como los satélites tipo *Cubesat*, los dispositivos programables como los FPGA y los SoC FPGA han adquirido relevancia debido a la flexibilidad que ofrecen para implementar y modificar funciones de *hardware* de acuerdo con las especificaciones técnicas de la misión [15]. No obstante, la operación de estos dispositivos depende de la información de configuración que define su funcionamiento, por lo que cualquier necesidad de actualización, restauración o recarga de dicha información obliga a contar con mecanismos de reconfiguración.

Bajo este panorama, en el presente capítulo se revisan proyectos y desarrollos relacionados con la reconfiguración de dispositivos FPGA y SoC FPGA, con el propósito de identificar sus principales características, alcances y limitaciones. A partir de ello, se busca establecer una comparación entre las soluciones encontradas e identificar áreas de oportunidad que permitan posicionar la propuesta desarrollada en esta tesis.

3.1. Proyectos relacionados con la reconfiguración de dispositivos FPGA y SoC FPGA

En esta sección se revisan trabajos y proyectos que guardan relación con la propuesta presentada en este trabajo. Como criterio principal, se priorizan aquellos proyectos que incorporan dispositivos FPGA o SoC FPGA en aplicaciones espaciales, puesto que el sistema propuesto está orientado a operar en dicho entorno. Asimismo, se consideran soluciones basadas en arquitecturas híbridas de procesamiento y capacidades de reconfiguración. A partir de esta revisión, se busca construir un panorama de trabajos relacionados que sirva como base para la comparación posterior.

3.1.1. Reconfiguración de alta confiabilidad en órbita para FPGA tipo *flash* en una misión de espacio profundo

Dispositivo de interés a reconfigurar

El trabajo presentado en [66] contempla la reconfiguración de múltiples dispositivos FPGA *Microchip* M2S090 basado en *flash*, seleccionados por su mayor resistencia frente a la inversión de bits causada por la radiación, en comparación con dispositivos basados en SRAM.

Aplicación

En una carga útil astrofísica integrada en el satélite *DRO-A*, el cual opera en la órbita retrógrada lejana (DRO, por sus siglás en inglés). Esta misión se desarrolla en un entorno espacial profundo con alta exposición a partículas de radiación, lo que impone condiciones estrictas sobre la selección de componentes electrónicos y en la confiabilidad del sistema propuesto.

Método de configuración

El sistema implementa la programación SPI en modo esclavo para la interacción entre el microcontrolador y el FPGA. En este caso, la reconfiguración es de tipo estática global (FSR, por sus siglas en inglés), por lo que no se contempla reconfiguración dinámica parcial (DPR, por sus siglas en inglés). La secuencia de programación se basa en la configuración del modo SPI, la transferencia del *bitstream* y la posterior verificación del proceso de escritura.

Uso de memoria y agente externo

La arquitectura emplea memoria externa NOR *flash* S25FL512S de 64 MB, perteneciente a la categoría COTS, para almacenar los paquetes de configuración de los FPGA. Dicha memoria desempeña una función equivalente a una copia de oro; es decir, almacena una versión de referencia del archivo de configuración, a partir de la cual puede realizarse nuevamente la configuración del FPGA cuando sea necesario. Su selección se debe a que presenta no volatilidad, capacidad de almacenamiento suficiente, simplicidad de integración y presenta alta tolerancia a la inversión de bits por efecto de la radiación. No obstante, el artículo no reporta datos cuantificables de tolerancia a la radiación para la memoria empleada.

La selección de esta memoria puede sustentarse adicionalmente en los resultados reportados en la tabla 2.1, donde se presenta una tolerancia a TID de $\approx 14{,}2\,\text{krad}$ y presentan un LETh $\approx 10\,\text{MeV}\cdot\text{cm}^2/\text{mg}$. Estos resultados aportan evidencia cuantitativa sobre su comportamiento frente a los efectos de radiación considerados en el estudio.

Como agente de control externo, se utiliza un microcontrolador basado en *Cortex*-M7 que, de acuerdo con su fabricante, entra en la categoría de tolerante a la radiación, con valores cuantificables reportados de TID $= 30\,\text{krad}$ y un LETh $= 60\,\text{MeV}\cdot\text{cm}^2/\text{mg}$. Encargado de gestionar la transferencia de datos y el proceso de reconfiguración.

Interfaz externa empleada en la reconfiguración

Se utiliza SPI de cuatro pines para la comunicación entre el microcontrolador y el FPGA de interés. En este caso, JTAG es utilizado para pruebas y depuración del sistema en tierra.

Aportación principal

- Arquitectura de reconfiguración orientada a la operación de múltiples FPGA en un entorno de espacio profundo.
- Uso de redundancia modular triple (TMR, por sus siglas en inglés) en el almacenamiento de datos para aumentar la confiabilidad del *bitstream*.
- Selección de memoria *flash* para el almacenamiento del *bitstream*, lo que prioriza retención de datos y tolerancia a la inversión de bits causada por efectos de la radiación.

3.1.2. Módulo de reconfiguración remota de FPGA para aplicaciones satelitales

Dispositivo de interés a reconfigurar

El trabajo presentado en [67] utiliza un FPGA *Altera Cyclone* IV EP4CE6E22C8N basado en SRAM, integrado en una carga útil de un satélite.

Aplicación

El sistema propuesto está pensado para satélites, más específicamente para *CubeSat*, dentro de una arquitectura de carga útil reconfigurable remotamente con el propósito de alargar la vida útil y actualizar funciones durante la misión.

Método de configuración

La reconfiguración se realiza por medio de un microcontrolador externo que recibe el archivo de configuración desde una estación terrestre, lo almacena en la memoria de configuración y luego reinicia el FPGA para que cargue su nueva configuración. En el artículo se indica que esto se hace con un *driver SRunner* modificado para que pueda ejecutarse en microcontroladores, lo que le permite usar archivos de datos de programación sin procesar (RPD, por sus siglás en inglés), y usar el protocolo SPI.

Uso de memoria y agente externo

El artículo menciona la implementación de un prototipo empleado para validar experimentalmente el funcionamiento del módulo de reconfiguración remota del FPGA, por lo que no se presentan datos cuantificables sobre tolerancia a la radiación en TID y LETh de los componentes seleccionados. La propuesta emplea una memoria externa tipo NOR *flash* EPCS16SI8N de *Altera* con capacidad de 2 MB, utilizada como copia de oro del *bitstream*. Aunque el artículo no reporta una caracterización frente a radiación de esta memoria, los

resultados recopilados en la tabla 2.1 para dispositivos de tecnología comparable proporcionan valores de referencia sobre su posible comportamiento. No obstante, estos valores no permiten establecer directamente la tolerancia a la radiación del dispositivo NOR *flash* EPCS16SI8N empleado en la propuesta.

Por otro lado, el microcontrolador empleado como agente externo para llevar a cabo la reconfiguración es un PIC18F6680 de *Microchip* no presenta, en el artículo o en su hoja de especificaciones técnicas, datos cuantificables de tolerancia a la radiación.

Interfaz externa empleada en la reconfiguración

Para la reconfiguración del FPGA, el microcontrolador utiliza una interfaz de configuración serial basada en SPI por medio del driver *SRunner*, con la que escribe el *bitstream* en la memoria externa y posteriormente controla las señales de configuración del FPGA, a fin de cargar la nueva imagen de configuración.

Aportación principal

- Propone un módulo de reconfiguración remota para FPGA en satélites.
- Implementan estrategias de mitigación de fallas en el controlador basadas en reserva en espera activa (WSP, por sus siglas en inglés).
- Contempla el uso del *bitstream* soportado por el dispositivo para proteger la información de configuración.

3.1.3. Reconfiguración de FPGA en operación para compresión hiperespectral flexible en un *CubeSat*

Dispositivo de interés a reconfigurar

El trabajo descrito en [68] presenta la reconfiguración de un SoC FPGA embebido en la unidad de procesamiento a bordo (OPU, por sus siglas en inglés), de los satélites *HYPSO-1 y HYPSO-2*. El artículo menciona que el OPU incorpora un dispositivo *Zynq* SoC de *AMD Xilinx* que contiene un procesador ARM que ejecuta *Linux* y una sección lógica programable basada en FPGA *Kintex*-7. En este trabajo, la reconfiguración se aplica sobre la zona PL del SoC FPGA.

Aplicación

La aplicación corresponde a un sistema de procesamiento a bordo para imágenes hiperesprectrales en satélites pequeños, especialmente *CubeSat*. El caso de estudio es el satélite *HYPSO-1*, un *CubeSat* de 6U con una cámara hiperespectral como carga útil, con el propósito de observar y analizar los océanos. La reconfiguración propuesta se usa para adaptar un acelerador de *hardware* de compresión de datos hiperespectrales según las dimensiones del volumen de datos capturados.

Método de configuración

El artículo menciona y compara dos métodos de reconfiguración, DPR y FSR. Aunque el FPGA a reconfigurar soporta ambas, eligieron como método por defecto FSR para simplificar el desarrollo y la operación satelital.

Uso de memoria y agente externo

El artículo no describe explícitamente una memoria externa dedicada a almacenar el archivo de reconfiguración, a diferencia de las anteriores arquitecturas, sino que aborda la reconfiguración desde el entorno del SoC, donde los *bitstream* son gestionados por el sistema operativo y sus herramientas de configuración. En este contexto, el *bitstream* de referencia empleado para restaurar o adaptar la configuración puede considerarse la copia de oro del sistema.

En cuanto al agente externo, en este caso no se trata de un microcontrolador supervisor/reconfigurador, más bien, se trata del *software* que se ejecuta en el procesador ARM del dispositivo *Zynq* SoC dentro del sistema *Linux* embebido del OPU, quien recibe comandos, verifica si existe un *bitstream* adecuado, y en caso de ser necesario ordena la reconfiguración del FPGA. Tanto el artículo como las hojas de especificaciones no reportan datos cuantificables sobre la tolerancia a la radiación, en términos de TID y LETh, del dispositivo *Zync* SoC.

Interfaz externa empleada en la reconfiguración

El artículo no menciona una interfaz física externa para la comunicación directa con el FPGA. Lo que se menciona es que el *software* interactúa con el FPGA vía memoria, lo que indica una integración dentro del SoC más que una reconfiguración por interfaz física externa como JTAG.

Aportación principal

- Demostrar la reconfiguración del FPGA en el satélite *HYPSO-1* en órbita durante operación.
- Definir un flujo de prueba y validación de nuevos diseños FPGA antes de integrarlos en operaciones rutinarias.
- Habilitar la compresión de *hardware* de múltiples dimensiones de imágenes hiperespectrales mediante reconfiguración automática.
- Reducir significativamente el tiempo de compresión de cubos hiperespectrales, pasando de $5-6$ min a menos de un segundo, gracias a la implementación de aceleradores en FPGA.

3.1.4. DPR de coprocesadores para procesamiento de imágenes en satélites pequeños

Dispositivo de interés a reconfigurar

La propuesta reportada en [8] contempla la región parcialmente reconfigurable dentro de un FPGA *Xilinx Artix*-7 200T. En dicha región se cargan distintos coprocesadores especializados, mientras que el resto del sistema permanece en operación.

Aplicación

El proyecto está orientado al procesamiento de imágenes a bordo de satélites pequeños. En particular, el proyecto emplea coprocesadores para realizar conversión rojo, verde, azul (RGB, por sus siglas en inglés) a escala de grises, detección de bordes por primera diferencia horizontal y detección de bordes por primera diferencia vertical. Con ello, se busca aumentar la capacidad de procesamiento de sistemas espaciales de baja potencia y con recursos limitados.

Método de configuración

El trabajo emplea DPR como método de configuración. El artículo remarca que pueden reconfigurarse regiones específicas del dispositivo sin interrumpir la operación del resto del sistema. De esta manera, es posible intercambiar distintos coprocesadores mientras el procesador principal continúa en funcionamiento.

Uso de memoria y agente externo

El proyecto emplea memoria externa para almacenar múltiples *bitstream* para DPR, lo que significa que guarda la copia de oro del sistema. Sin embargo, el artículo no especifica el modelo exacto de la memoria utilizada ni reporta datos cuantificables sobre su tolerancia a la radiación en términos de TID y LETh.

Asimismo, el sistema utiliza un microcontrolador externo para controlar el proceso de reconfiguración del FPGA y automatizar el intercambio de coprocesadores. No obstante, el artículo tampoco proporciona el modelo exacto empleado ni reporta información cuantificable sobre la tolerancia a la radiación del dispositivo en términos de TID y LETh.

Interfaz externa empleada en la reconfiguración

La reconfiguración parcial se realiza por medio de la interfaz JTAG. El microcontrolador externo controla este proceso a través de *software* que genera las señales necesarias para llevar a cabo la reconfiguración parcial del dispositivo, en una implementación de tipo *bit-banging* (JTAG emulado por GPIO). Además, la plataforma incluye soporte para SPI y UART, utilizados para funciones auxiliares como monitorear el sistema, almacenamiento de *bitstream* y comunicación, entre otras.

Aportación principal

- Demostrar una forma de intercambiar coprocesadores mediante DPR.

- Automatizar el proceso de intercambio por medio de un microcontrolador externo, encargado de gestionar la reconfiguración del dispositivo.
- Implementar y validar coprocesadores de imagen en un sistema de bajo consumo para satélites pequeños.
- Emulación del estándar JTAG por medio de *bit-banging*.
- Demostrar que la reconfiguración es una forma de depurar el *hardware* del sistema de efectos de evento único (SEE, por sus siglas en inglés), al cargar un nuevo coprocesador. Asi se aporta un beneficio limitado de recuperación frente a la radiación en el subsistema del coprocesador.

3.1.5. Unidad de procesamiento de datos reconfigurable tipo SoC para instrumentación espacial

Dispositivo de interés a reconfigurar

La arquitectura desarrollada en [69] emplea la reconfiguración de una unidad de procesamiento de datos reconfigurables basada en FPGA, bajo un enfoque SoC para instrumentación espacial. El artículo presenta arquitecturas sustentadas en FPGA *Xilinx Virtex*-I y *Virtex*-II, dentro de las cuales se integran el procesador, funciones específicas de procesamiento y las interfaces principales del sistema. Asimismo, se presenta una variante más avanzada basada en *Virtex*-II Pro, que cuenta con procesadores PowerPC 405 embebidos en el dispositivo.

Aplicación

La propuesta está orientada a instrumentos espaciales de obtención y procesamiento de imágenes, donde las unidades de procesamiento de datos (DPU, por sus siglas en inglés) actúan como interfaz entre la nave y los sensores del instrumento. Por lo que, la arquitectura reconfigurable se emplea para realizar tareas de control, procesamiento específico de datos científicos, formateo/codificación y compresión de imágenes, particularmente en instrumentos que demandan altas tasas de obtención y procesamiento. El artículo hace referencia a la experiencia adquirida con la cámara de monitoreo *express* (VMC, por sus siglas en inglés), que a partir de ella, deriva una propuesta de DPU configurable para futuros sensores avanzados de imágenes.

Método de configuración

El método de reconfiguración tomado en cuenta en este trabajo se trata de FSR y DPR. Este enfoque permite modificar la funcionalidad del sistema tanto durante la etapa de desarrollo como en vuelo. Para la variante basada en *Virtex*-II, la reprogramación en órbita se logra gracias a la implementación de fuentes de almacenamiento externas de *Xilinx*, que pueden ser programadas remotamente desde la nave a través de una lógica implementada en un dispositivo supervisor. Este esquema permite la actualización del *hardware* y el reemplazo o incorporación de módulos funcionales según las necesidades de la misión.

Uso de memoria y agente externo

La propuesta incorpora diversas memorias como SRAM, SDRAM, DRAM, PROM + EEPROM externas pertenecientes a *Xilinx* para soportar la operación de la unidad de procesamiento. En la variante basada en *Virtex*-II, la reconfiguración en vuelo se logra mediante EEPROM de *Xilinx*, las cuales contienen la copia de oro del sistema y pueden ser actualizadas remotamente desde la nave. No obstante, el artículo no proporciona valores cuantificables de tolerancia a la radiación en términos de TID y LETh de la memoria empleada. Como referencia, los resultados recopilados en la tabla 2.1 para dispositivos EEPROM COTS caracterizados frente a radiación reportan valores de TID = 90 krad y LETh= $10-20\,\mathrm{MeV}\cdot\mathrm{cm}^2/\mathrm{mg}$. Estos valores permiten establecer una referencia para esta tecnología, pero no determinan directamente la tolerancia a la radiación del dispositivo empleado en la propuesta.

En cuanto al agente externo de supervisión/configuración, el sistema emplea un FPGA *Actel* versión *rad-hard*, encargada de funciones como el control de configuración de los FPGA, la conmutación entre memorias de configuración, la supervisión de señales críticas, el control de *latch-up* y la activación de reinicio por *watchdog* en caso de falla. Aunque el artículo no especifica el modelo exacto del dispositivo, familias *rad-hard* de *Actel/Microchip* empleadas en aplicaciones espaciales, como *RTAX-S*, reportan valores del orden de TID = 300 krad y un LETh $> 117\,\mathrm{MeV}\cdot\mathrm{cm}^2/\mathrm{mg}$ [70], por lo que se puede decir que representa una referencia válida del nivel de endurecimiento frente a la radiación que puede ofrecer este tipo de supervisor.

Interfaz externa empleada en la reconfiguración

Para la reconfiguración en vuelo, el sistema emplea JTAG mediante la cual el FPGA *Actel* supervisor reconfigura remotamente las memorias EEPROM de configuración correspondientes a los FPGA.

Aportación principal

- Desarrollo de una DPU reconfigurable tipo SoC para instrumentación espacial.
- Integración de procesador, compresión, interfaces y lógica específica de procesamiento en el FPGA.
- Implementación de una estrategia de reprogramación en vuelo mediante EEPROM de configuración.
- Uso de un FPGA *rad-hard* como supervisión para control de configuración y tolerancia a la radiación.

3.1.6. Plataforma SoC basada en FPGA para cómputo espacial reconfigurable

Dispositivo de interés a reconfigurar

El trabajo presentado en [71] contempla la reconfiguración de una plataforma SoC implementada en una FPGA, en el cual se integran procesador, memoria, periféricos y bloques de *hardware* especializados en un mismo circuito programable. El artículo aborda este tipo de plataforma como una solución reconfigurable para aplicaciones espaciales, ya que destaca que el dispositivo permite incorporar tanto procesadores embebidos como coprocesadores según las especificaciones técnicas de la misión.

Aplicación

La propuesta está orientada a cómputo espacial y procesamiento de datos a bordo, especialmente en instrumentación compatible con aplicaciones espaciales. El artículo menciona procesamiento de señales, comunicación inalámbrica, multimedia y computación avanzada. Cada uno de ellos relevantes para sistemas embebidos reconfigurables en el sector espacial. Asimismo, plantea que una plataforma SoC FPGA puede emplearse en pequeñas naves espaciales como una solución genérica de cómputo reconfigurable remotamente desde la estación terrena.

Método de configuración

El artículo plantea usar DPR, ya que este método ofrece reconfigurar ciertas zonas específicas del dispositivo sin la necesidad de perder tiempo en una reconfiguración FSR.

Uso de memoria y agente externo

La propuesta considera el empleo de memoria externa para almacenar la copia de oro del sistema, destinada a la inicialización del programa del procesador embebido y a la operación general del sistema SoC. Sin embargo, el artículo no especifica modelos exactos de memoria ni presenta datos cuantificables de tolerancia a la radiación, tales como TID y LETh, para los dispositivos de memoria considerados.

Para el caso de un agente externo, aunque el trabajo menciona la posibilidad de reconfiguración remota desde estación terrena, no describe un agente externo dedicado, como un microcontrolador o supervisor, encargado de gestionar la reconfiguración del sistema.

Interfaz externa empleada en la reconfiguración

El artículo menciona la interfaz JTAG para depuración y descarga del *bitstream* hacia el FPGA.

Aportación principal

- Destaca la integración dentro de un mismo sistema programable, de procesador embebido, memoria, periféricos y aceleradores de *hardware*.

- Capacidad de incorporar coprocesadores o aceleradores específicos para resolver tareas computacionales intensivas.
- Expone distintas alternativas de interfaz entre procesador y aceleradores, así se demuestra flexibilidad en la integración de *hardware* especializado.

3.1.7. Computadora central de a bordo reconfigurable implementada mediante FPGA basada en SRAM y *flash* para aplicaciones espaciales

Dispositivo de interés a reconfigurar

En [9], el objetivo es un sistema de computo central a bordo del satélite *Flying laptop*, implementado en su totalidad sobre una arquitectura basada en FPGA. Particularmente, la tesis propone una computadora de a bordo compuesto por cuatro nodos centrales de procesamiento basados en FPGA *Xilinx Virtex*-II Pro XC2VP50 basado en SRAM, más un bloque de decodificación y votación implementado mediante FPGA basado en *flash*. Por lo que, el elemento de interés no es solo un FPGA aislado, sino una arquitectura compleja de una computadora de a bordo reconfigurable.

Aplicación

La propuesta se aplica a una computadora de a bordo de un pequeño satélite llamado *Flying laptop* para propósitos de pruebas tecnológicas. Esta computadora debe controlar los subsistemas y sus respectivas operaciones dentro del satélite, eso incluye procesamiento de imágenes, determinación y control de actitud, telecomunicaciones y manejo de datos, entre otras funciones.

La tesis se orienta a demostrar el uso de tecnologías FPGA aplicadas a una misión espacial real de órbita baja, con el objetivo de validar el concepto de computadora reconfigurable aplicada en una computadora de a bordo.

Método de configuración

La idea central consiste en aplicar tanto FSR como DPR en los dispositivos empleados como nodos de procesamiento, así como en el dispositivo destinado como elemento supervisor y de votación. Esto con el propósito de prevenir SEE mediante la reconfiguración periódica, de modo que la arquitectura funcione como un sistema redundante y reconfigurable.

Uso de memoria y agente externo

La propuesta incorpora memorias NOR *flash* para almacenar la copia de oro del sistema y así soportar la operación de la computadora de a bordo; sin embargo, la tesis no indica el modelo empleado ni reporta sus parámetros cuantificables de tolerancia a la radiación en términos de TID o LETh. Como referencia, los resultados recopilados en la tabla 2.1 para memorias NOR *flash* COTS caracterizadas frente a radiación reportan valores de TID $\approx$ 14,2 krad y de LETh $\approx$ 10 MeV $\cdot$ cm^2/mg. Estos resultados propor-

cionan una referencia sobre el comportamiento observado en dispositivos con tecnología comparable, pero no permiten establecer directamente la tolerancia a la radiación de la memoria empleada en la propuesta.

En cuanto al agente externo de supervisión y votación se emplea un FPGA *Actel RTProASIC* basado en *flash*, seleccionado por presentar mayor tolerancia frente a ciertos efectos radioactivos. Para esta familia, sí se reportan datos cuantificables de tolerancia a la radiación en términos TID $\approx$ 25 krad; sin embargo, la tesis y la hoja de especificaciones del dispositivo no cuentan con el dato cuantificable de LETh, por lo que se toma un valor aceptable de una presentación oficial de *Microsemi*, la cual reporta un valor de LETh $= 68\,\mathrm{MeV} \cdot \mathrm{cm}^2/\mathrm{mg}$ para esta familia de dispositivos [72].

Interfaz externa empleada en la reconfiguración

Para la reconfiguración y supervisión de los FPGA a reconfigurar, se hace énfasis en la importancia de JTAG como interfaz de configuración para los FPGA.

Aportación principal

- Propone una arquitectura para una computadora de a bordo reconfigurable basada completamente en FPGA con aplicaciones espaciales.
- Introduce una estrategia de mitigación de fallas basada en el uso combinado de FPGA basados en SRAM y *flash*.
- Aplicación en una misión espacial real con propósito de demostración de tecnología.
- Implementa el concepto de cuatro nodos de procesamiento FPGA basados en SRAM y un supervisor/votante FPGA basado en *flash*, como una computadora central de satélite.
- Desarrolla un entorno de *hardware-in-the-loop* para validar el funcionamiento de la computadora de a bordo y sus algoritmos de control.

3.2. Análisis comparativo de las soluciones reportadas y áreas de oportunidad

La revisión realizada muestra que la implementación de plataformas PLD y SoC FPGA han sido empleadas en aplicaciones espaciales debido, entre otras características, a su capacidad de reconfiguración. En los trabajos analizados, esta capacidad se utiliza para modificar funciones del sistema, adaptar el *hardware* a distintos objetivos de la misión y recuperar su operación ante determinados eventos, como los SEE. De esta manera, la reconfiguración constituye una estrategia de interés para el desarrollo de sistemas electrónicos destinados al entorno espacial.

Los trabajos analizados emplean distintos enfoques para llevar a cabo la reconfiguración del dispositivo de interés, estos incluyen reprogramación remota en órbita, DPR, arquitecturas redundantes con votación y plataformas SoC orientadas al procesamiento de a bordo.

Asimismo, se identificó que la memoria externa representa un elemento central dentro del proceso de reconfiguración, ya que en varios casos es el encargado de almacenar la copia de oro del sistema. Del mismo modo, el control de la reconfiguración suele realizarse por medio de un agente externo, como por ejemplo, un microcontrolador supervisor/reconfigurador, un FPGA supervisor/reconfigurador o por el propio sistema SoC. Entre las interfaces más empleadas para llevar a cabo la reconfiguración se encuentran JTAG y SPI.

Sin embargo, ninguno de los proyectos reportados coincide de manera exacta con la propuesta de este trabajo, ya que en muchos casos se enfocan más en la capacidad de procesamiento o en validar plataformas reconfigurables completas. Además, no todos los trabajos reportan parámetros cuantificables de tolerancia a la radiación para las memorias y agentes supervisores/reconfiguradores. Por ello, esta revisión permite identificar las bases técnicas ya existentes como los vacíos que justifican el desarrollo de la propuesta presentada en este trabajo.

A partir de los proyectos revisados en la sección 3.1, se presenta la tabla comparativa 3.1, con el fin de identificar sus características principales y sus diferencias con respecto a la propuesta de esta tesis.

Atributos / Proyectos	**Dispositivo de interés a reconfigurar**	**Aplicación**	**Método de configuración**	**Uso de memoria y agente externo**	**Interfaz externa empleada en la reconfiguración**	**Aportación principal**	**Limitaciones respecto a la propuesta**
[66]	Mútiples FPGA *Microchip* M2S090 de tipo *flash*	Carga útil astrofísica en el satélite *DRO-A* en el espacio profundo	Reconfiguración estática global	Memoria externa NOR *flash* S25FL512S Agente externo microcontrolador SAMV71 basado en *Cortex*-M7	SPI modo esclavo	Arquitectura de reconfiguración orientada a espacio profundo con almacenamiento redundante	Sin comparación de memorias; reconfiguración lenta ($20 - 30$ min)
[67]	FPGA *Altera Cyclone* IV	Validación experimental de un módulo de reconfiguración remota para satélites	Reconfiguración remota de la FPGA	Memoria externa NOR *flash* EPCS16SI8N Agente externo microcontrolador PIC18F6680	SPI por medio del *driver SRunner*	Demuestra un módulo funcional de reconfiguración remota orientado a satélites	Análisis radiactivo limitado; alto consumo de tiempo/energía; depende de ventana de enlace
[68]	SoC FPGA *Zynq*	Compresión hiperespectral flexible durante operación del satélite	Reconfiguración estática global y dinámica parcial	No contempla memoria ni agente externo	Sin interfaz externa para la reconfiguración	Demuestra reconfiguración en tiempo real, para adaptar algoritmos de compresión	Sin memoria externa; enfocado a flexibilidad; sin recuperación por radiación; análisis radiativo limitado
[8]	FPGA *Artix*-7 200T	Procesamiento de imágenes a bordo de satélites pequeños	Reconfiguración dinámica parcial	Si contempla memoria externa, pero no indica el modelo exacto Si contempla un agente externo microcontrolador, pero no proporciona el modelo exacto	JTAG emulado por GPIO	Demuestra intercambio dinámico de coprocesadores sin detener el sistema principal	Solo reconfiguración parcial; tiempo de reconfiguración alto
[69]	DPU reconfigurable tipo SoC basado en varios modelos de FPGA *Virtex* (I, II, II-Pro)	Instrumentos espaciales para imagen y procesamiento de datos científicos	Reprogramación global o parcial de forma remota si así se desea	EEPROM de configuración agente externo FPGA *Actel rad-hard*	JTAG	Propone una DPU reconfigurable para instrumentación espacial con supervisor y actualización remota en vuelo	Arquitectura compleja; análisis radiativo limitado; costoso
[71]	Plataforma SoC basada en FPGA	Computo espacial y procesamiento de datos a bordo	Reconfiguración dinámica parcial	Sí contempla memoria externa, pero no específica modelo; no contempla agente externo	JTAG	Justifica el uso de SoC basados en FPGA para cómputo espacial flexible	Enfoque conceptual; sin agente externo claro; pocos datos radiactivos
[9]	Múltiples FPGA *Virtex*-II Pro XC2VP50 basados en SRAM	Computadora central del satélite *Flying laptop*	Reconfiguración global o parcial	Memoria externa NOR *flash*, pero no especifica modelo agente externo FPGA *Actel RTProASIC* basado en *flash*	JTAG	Propone una OBC espacial completamente basada en FPGA con mitigación de radiación por combinación SRAM/*flash*	Enfoque redundante; alta complejidad; alto costo

Tabla 3.1: Comparación de soluciones reportadas y áreas de oportunidad. Elaboración propia con base en: [66, 67, 68, 8, 69, 70, 71, 9, 72].

A partir del análisis comparativo de las soluciones reportadas, se observa que, aunque los trabajos revisados demuestran viabilidad de la reconfiguración de dispositivos FPGA y SoC FPGA en aplicaciones espaciales, en la mayoría de los casos el enfoque se orienta hacía la flexibilidad funcional del sistema, el incremento de la capacidad de procesamiento o la implementación de arquitecturas redundantes complejas. No obstante, se identifican vacíos importantes en propuestas específicamente dirigidas al desarrollo de un sistema de reconfiguración dedicado para recuperar la funcionalidad de un dispositivo maestro ante

fallas asociadas a SEE, así como la justificación técnica de la memoria de almacenamiento y el equilibrio entre simplicidad de integración y costo. Por lo que, las áreas de oportunidad identificadas permiten definir con mayor claridad los elementos que la propuesta de este trabajo puede aportar frente a los antecedentes revisados:

- Sistema de reconfiguración orientado a la recuperación de la configuración de la región PL del SoC FPGA ante alteraciones de la memoria de configuración producidas por SEU, así como ante SEFI cuya recuperación pueda realizarse mediante la recarga del *bitstream* de referencia.
- Selección de la tecnología de almacenamiento fundamentada en una investigación comparativa (tabla 2.4.3).
- Empleo de componentes COTS técnicamente justificada, que permita reducir costos sin perder de vista los efectos del entorno espacial. Además, reduce la dependencia de infraestructuras complejas o componentes costosos como los de grado militar/espacial.
- Uso de una arquitectura de *hardware* con menor complejidad, en comparación con soluciones redundantes o plataformas completas de cómputo reconfigurable.

La propuesta se sustenta en la selección de tecnologías de almacenamiento fundamentada en la subsección 2.4.3, en la aplicación de componentes COTS y en una arquitectura enfocada principalmente en la recuperación de la configuración ante SEE no destructivos. En particular, se consideran las alteraciones de bits en la memoria de configuración producidas por SEU, las cuales pueden modificar el comportamiento de la lógica programable, así como aquellos SEFI cuya recuperación sea posible mediante la reconfiguración del dispositivo [73]. En ambos casos, la estrategia de recuperación considerada consiste en la recarga del *bitstream* de referencia para restablecer la configuración del dispositivo. De esta manera, la propuesta se orienta a la recuperación de la configuración del dispositivo, más que a la modificación dinámica funcional.

Capítulo 4

Metodología de diseño del sistema de reconfiguración

En este capítulo se presenta la metodología de diseño empleada para el desarrollo del sistema de reconfiguración propuesto para un SoC FPGA *Xilinx Zynq*-7000 (XC7Z020), basado en un microcontrolador como gestor del proceso. El enfoque adoptado se fundamenta en la metodología de diseño de *Ulrich* [74], la cual organiza el desarrollo en fases sucesivas que permiten avanzar de manera ordenada desde la planeación hasta el inicio de producción del sistema. Sin embargo, de acuerdo con los objetivos del trabajo (sección 1.3), se emplea una adaptación de esta metodología, donde se omiten las etapas de planeación e inicio de producción, lo que permite concentrarse en las fases de desarrollo del concepto, diseño a nivel de sistema, diseño de detalle, y pruebas y refinamiento. A lo largo de este capítulo se describen las etapas, desde el análisis del problema y la definición de requisitos, hasta la estructuración de la arquitectura general y la selección de componentes.

4.1. Metodología de diseño adoptada

Como se mencionó, la metodología de diseño de Ulrich [74] es la empleada en el desarrollo del sistema de reconfiguración; sin embargo, debido a los objetivos y alcances del trabajo, se emplea una adaptación de esta, la cual organiza el proceso en una serie de etapas que abarcan desde la definición conceptual hasta la verificación funcional del producto. Esta metodología se selecciona debido a que permite estructurar de manera sistemática el diseño de sistemas de ingeniería, así se facilita la identificación de requisitos, la arquitectura del sistema, y su implementación. Por lo tanto, el presente trabajo se desarrolla a través de las siguientes etapas:

- Desarrollo del concepto: identificación de las necesidades del cliente, generación de ideas del producto, análisis de la competencia y selección del concepto viable con justificación económica, lo que da como resultado el listado de requisitos del proyecto.

- Diseño a nivel de sistema: definición de la arquitectura general del producto, descomposición en subsistemas y elaboración de diagramas de bloques con requisitos.
- Diseño de detalle: selección de materiales, componentes y circuitos, así como diseño de algoritmos que satisfacen las especificaciones técnicas definidas en la etapa previa de diseño.
- Pruebas y refinamiento: construcción de prototipos de preproducción, verificación funcional del desempeño frente a los requisitos y ajustes necesarios para asegurar que el producto cumpla con lo esperado.

En la figura 4.1 se muestra un diagrama de la adaptación de esta metodología para el sistema propuesto. Cada una de estas etapas se desarrollan con más detalle en los capítulos posteriores.

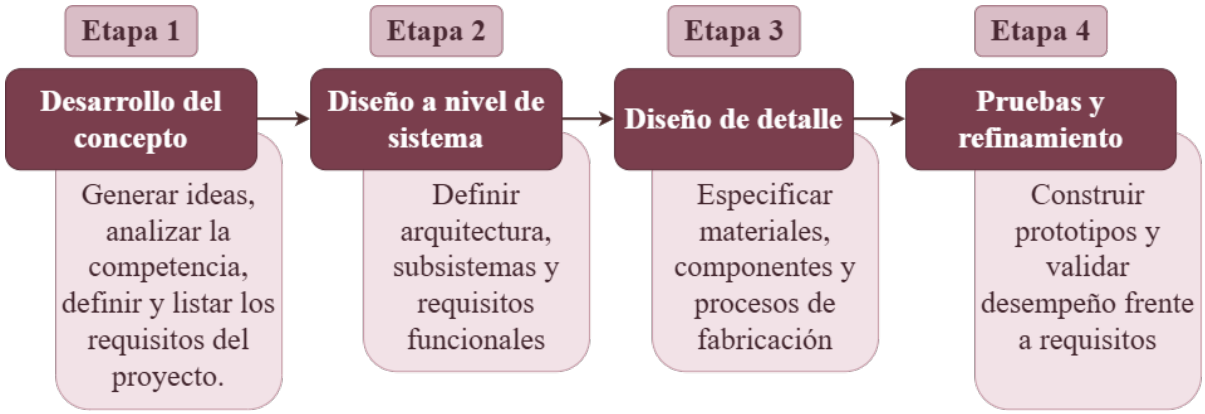


Figura 4.1: Diagrama de metodología de diseño de Ulrich adaptada para el sistema propuesto. Fuente: [74].

Capítulo 5

Diseño

En este capítulo se presenta el diseño del sistema, estructurado en tres etapas principales: diseño conceptual, diseño a nivel de sistema y diseño de detalle. Con el esquema anterior, se puede abordar de manera progresiva y ordenada la definición del prototipo de reconfiguración del SoC FPGA *Xilinx Zynq*-7000 (XC7Z020). El diseño conceptual establece el panorama general del sistema, los propósitos a cumplir y sus alcances, lo que permite identificar las funciones principales y las necesidades que debe satisfacer. Posteriormente, el diseño a nivel de sistema se enfoca en la definición de la arquitectura, determinación de especificaciones técnicas y la interacción entre subsistemas. Por último, el diseño de detalle realiza la selección de tecnología, circuitos, desarrolla el *software* con el fin de cumplir con las especificaciones definidas en la etapa anterior de diseño. De esta manera, el capítulo presenta un marco metodológico sólido que asegura que las opciones finales en cada etapa de diseño contribuyan al desempeño del sistema.

5.1. Objetivo del sistema de reconfiguración

El sistema tiene como objetivo realizar la reconfiguración de la zona PL de un dispositivo SoC FPGA *Xilinx Zynq* (XC7Z020) por medio de la obtención y transmisión del *bitstream*, así como la ejecución de las operaciones de control necesarias para la selección del modo de arranque y la inicialización del dispositivo. La información de configuración se transmite hacia el dispositivo de interés a través de una interfaz JTAG emulada, lo que garantiza un proceso de configuración controlado.

5.2. Diseño conceptual

El diseño conceptual del sistema de reconfiguración tiene como objetivo dar a conocer las bases generales del proyecto, desde las actividades que debe realizar y bajo que requisitos debe operar, sin abordar aún detalles de implementación ni características técnicas de los componentes. En este nivel de abstracción de diseño se describe, de manera general, como está compuesto el sistema y los requisitos que rigen su operación, de acuerdo con el

objetivo principal. Además, este apartado permite delimitar sus alcances y guiar con una mejor claridad las decisiones de diseño en los siguientes niveles de abstracción.

5.2.1. Descripción general del sistema de reconfiguración

El sistema de reconfiguración tiene como propósito restablecer la reconfiguración de un SoC FPGA *Xilinx Zynq* (XC7Z020), cuando sea requerido. Para ello, el bitstream es transmitido a través de la interfaz JTAG. Este proceso permite cargar nuevamente la configuración de la lógica programable y puede emplearse como mecanismo de recuperación ante alteraciones de la memoria de configuración producidas por SEU, así como ante aquellos SEFI cuya recuperación pueda realizarse mediante la reconfiguración del dispositivo.

Para poder cubrir con las necesidades, se puede definir la composición del sistema por medio de dos elementos principales: una fuente de almacenamiento externa para los *bitstream* y un dispositivo reconfigurador, ambos con una tolerancia a los efectos de la radiación significativamente alta. La fuente de almacenamiento es la encargada de resguardar el archivo de reconfiguración. El dispositivo reconfigurador concentra la mayor parte de las operaciones del sistema, entre las cuales se incluyen la ejecución de comandos de inicialización del SoC FPGA, la obtención del *bitstream* y su envío al dispositivo a reconfigurar por medio del estándar JTAG.

El dispositivo a reconfigurar corresponde a un SoC FPGA *Xilinx Zynq* (XC7Z020). Este es el elemento externo de interés sobre el cual actúa el sistema de reconfiguración. Es quien recibe la información binaria transmitida y ejecuta las funciones establecidas una vez que concluye el proceso de configuración.

A continuación, se presenta un diagrama de bloques correspondiente a esta primera etapa de diseño del sistema de reconfiguración.

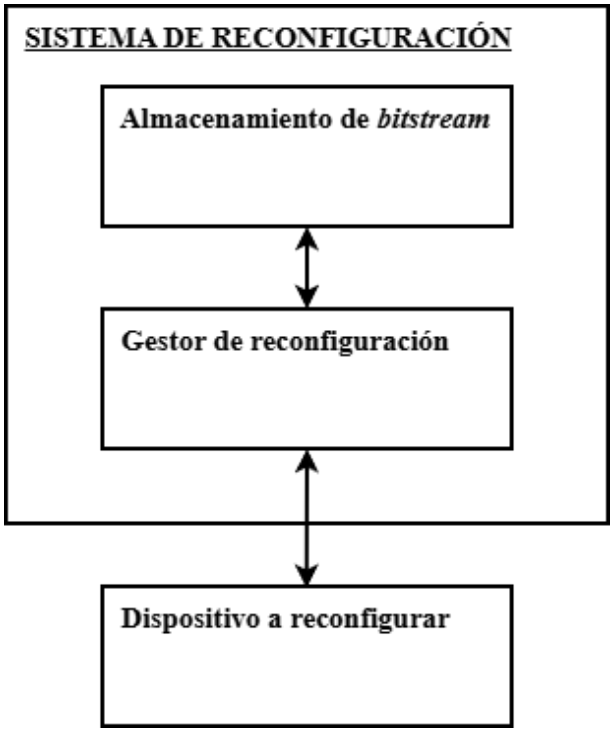


Figura 5.1: Diagrama de bloques del sistema de reconfiguración.
Etapa de diseño conceptual.

5.2.2. Requisitos del sistema de reconfiguración

El sistema debe ser capaz de cumplir con los siguientes requisitos para satisfacer las necesidades del proyecto.

- El sistema debe ser capaz de almacenar el *bitstream* y gestionar su flujo desde su obtención hasta su transmisión hacia el SoC FPGA *Xilinx Zynq*-7000 (XC7Z020).
- El sistema debe ser capaz de preservar la integridad del *bitstream* durante su acceso, manejo y transmisión, con el fin de evitar configuraciones incompletas o erróneas.
- El sistema debe ser capaz de modificar el modo de reconfiguración del SoC FPGA *Xilinx Zynq* (XC7Z020).
- El sistema debe ser capaz de establecer comunicación con el SoC FPGA *Xilinx Zynq*-7000 (XC7Z020) por medio del estándar JTAG para lograr su correcta inicialización y reconfiguración.
- El sistema debe ejecutar el proceso de reconfiguración de manera controlada, respetando la secuencia de instrucciones y transferencia de datos requeridas por el dispositivo.
- Tanto el sistema como el algoritmo desarrollado deben ser compatibles con el SoC FPGA *Xilinx Zynq*-7000 (XC7Z020).

5.2.3. Herramienta auxiliar para el análisis del bitstream

Como parte del diseño conceptual del sistema de reconfiguración, se identifica la necesidad de contar con una herramienta auxiliar que permite analizar la estructura y organización de la información contenida en el *bitstream*. Debido a que este archivo contiene la información de configuración que posteriormente será transmitida hacia la lógica programable del SoC FPGA, resulta necesario comprender los elementos que lo conforman y la secuencia en la que se encuentran organizados.

Por esta razón, se plantea el desarrollo de una herramienta auxiliar para el análisis del *bitstream*, cuyo propósito es facilitar la identificación de elementos como encabezados, comandos, registros y datos de configuración. La información obtenida mediante esta herramienta servirá como apoyo para comprender la secuencia de configuración y para el posterior desarrollo del algoritmo encargado de transmitir el *bitstream* hacia el dispositivo.

Cabe señalar que esta herramienta no forma parte de la arquitectura final del sistema de reconfiguración, sino que se considera un recurso de apoyo para su proceso de diseño. Su desarrollo e implementación se presentan posteriormente en la subsección 5.4.5.

5.3. Diseño a nivel de sistema

En esta etapa de diseño se establece la estructura general del sistema, lo que permite definir su arquitectura, bloques principales y la interacción entre ellos, con el fin de asegurar coherencia entre los requisitos planteados (subsección 5.2.2) y su futura implementación. Asimismo, se identifican las especificaciones técnicas del sistema, lo que facilita la toma de decisiones arquitectónicas y de selección de componentes antes de entrar en la etapa de diseño de detalle. De este modo, se reducen los riesgos y se garantiza una integración adecuada de todos los componentes. Finalmente, en esta etapa se presenta el diagrama de estados que describe el comportamiento general del sistema.

5.3.1. Arquitectura general del sistema de reconfiguración

La arquitectura se organiza de manera jerárquica, donde el dispositivo reconfigurador actúa como el elemento central de control. Este dispositivo coordina el proceso de reconfiguración y obtiene el *bitstream* desde la fuente de almacenamiento externa cuando se requiere modificar o restablecer la configuración del dispositivo. La configuración puede realizarse para actualizar la funcionalidad de acuerdo con los objetivos de la misión, o para recuperar la configuración de la lógica programable ante una alteración previamente detectada por un mecanismo externo. En este último caso, el sistema propuesto actúa sobre la etapa de recuperación mediante la carga del *bitstream* correspondiente.

El dispositivo reconfigurador transmite la señal para seleccionar el modo de arranque del SoC FPGA. Posteriormente, mediante la interfaz JTAG, envía los comandos de inicialización para indicar que el proceso de configuración está a punto de comenzar. Una vez establecida la secuencia de arranque, el *bitstream* es obtenido por el dispositivo re-

configurador y transmitido hacía el SoC FPGA, de modo que, al recibir la información de configuración, este ejecute las tareas que le fueron asignadas.

5.3.2. Identificación de subsistemas e interfaces de comunicación

El sistema se puede fragmentar en dos subsistemas: almacenamiento y control. Cada uno de estos cumple con una función específica dentro de la arquitectura general del sistema, lo que permite una separación clara de responsabilidades.

Subsistema de almacenamiento

Es el encargado de resguardar el *bitstream* contra los efectos de radiación espacial. Para lograrlo, debe contar con las siguientes especificaciones técnicas:

- Componente perteneciente a la categoria COTS: se prioriza la selección de una memoria comercial de tipo COTS, debido a su disponibilidad y menor costo respecto a componentes específicamente calificados para aplicaciones espaciales [75].
- Memoria no volátil con tecnología FRAM/MRAM: de acuerdo con la investigación presentada en la subsección 2.4.1, se prioriza la selección de memorias no volátiles basadas en tecnologías FRAM/MRAM debido a sus mecanismos físicos de almacenamiento. En las memorias FRAM, la información se conserva mediante la polarización de un material ferroeléctrico, mientras que en las MRAM se representa mediante estados magnéticos. A diferencia de tecnologías cuyo almacenamiento depende de carga eléctrica, estos mecanismos presentan menor susceptibilidad a errores blandos inducidos por radiación, particularmente a inversiones de bits asociadas a SEU [76, 63, 77].
- Capacidad de almacenamiento $\geq 4\,\text{MB}$: La memoria debe contar con una capacidad igual o superior a 4 MB ($\approx 4\,045\,564$ bytes). Este valor se obtuvo de manera experimental mediante la herramienta *Vivado*, como se muestra en la figura 5.2 y corresponde con el tamaño esperado del *bitstream*, de acuerdo con la estructura descrita en el manual UG470 [19].

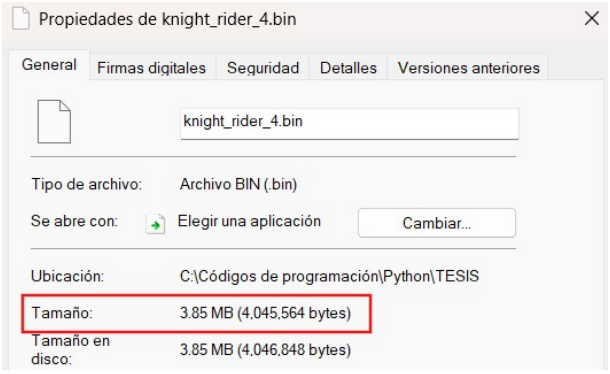


Figura 5.2: Tamaño del *bitstream* de un proyecto sintetizado en *Vivado*.

- Interfaz de comunicación serial: dado que el *bitstream* debe transferirse desde la memoria hacia el dispositivo reconfigurador, se requiere de una interfaz que permita realizar esta transferencia mediante un número reducido de señales y con una velocidad de operación adecuada para el proceso de reconfiguración. Como referencia, el puerto TAP del SoC FGPA admite una frecuencia máxima de TCK de 66 MHz, como se muestra en la figura 5.3; sin embargo, este valor corresponde al límite de operación del dispositivo y no establece directamente la frecuencia requerida para la interfaz de la memoria.

Configuration Switching Characteristics

Table 101: **Configuration Switching Characteristics**

Symbol	Description	Speed Grade				Units
		-3	-2	-1C/-1I/-1LI	-1Q	
Power-up Timing Characteristics						
T_{PL} (1)	Program latency	5.00	5.00	5.00	5.00	ms, Max
T_{POR}	Power-on reset (50 ms ramp rate time)	10/50	10/50	10/50	10/50	ms, Min/Max
	Power-on reset (1 ms ramp rate time) with the power-on reset override function disabled; (devcfg.CTRL.PCFG_POR_CNT_4K = 0).(2)	10/35	10/35	10/35	10/35	ms, Min/Max
	Power-on reset (1 ms ramp rate time) with the power-on reset override function enabled; (devcfg.CTRL.PCFG_POR_CNT_4K = 1).(2)	2/8	2/8	2/8	2/8	ms, Min/Max
$T_{PROGRAM}$	Program pulse width	250.00	250.00	250.00	250.00	ns, Min
Boundary-Scan Port Timing Specifications						
T_{TAPTCK}/T_{TCKTAP}	TMS and TDI setup/hold	3.00/2.00	3.00/2.00	3.00/2.00	3.00/2.00	ns, Min
T_{TCKTDO}	TCK falling edge to TDO output	7.00	7.00	7.00	7.00	ns, Max
F_{TCK}	TCK frequency	66.00	66.00	66.00	66.00	MHz, Max
Internal Configuration Access Port						
F_{ICAPCK}	Internal configuration access port (ICAPE2)	100.00	100.00	100.00	100.00	MHz, Max
Device DNA Access Port						
F_{DNACK}	DNA access port (DNA_PORT)	100.00	100.00	100.00	100.00	MHz, Max

Figura 5.3: Frecuencia máxima de TCK para dispositivos SoC FPGA de *Xilinx*.
Fuente: [78]

La selección de la interfaz de comunicación no se establece únicamente a partir de su frecuencia máxima de operación. Las interfaces paralelas pueden proporcionar mayores tasas de transferencia debido a que transmiten varios bits simultáneamente; sin embargo, requieren líneas independientes para datos, direccionamiento y control. En contraste, las interfaces seriales permiten realizar la transferencia mediante un número menor de señales, reduciendo la cantidad de conexiones necesarias entre la memoria y el dispositivo reconfigurador [79, 80].

Para la arquitectura propuesta se prioriza una interfaz serial, ya que permite reducir el número de señales destinadas a la comunicación con la memoria sin descartar velocidades de transferencia adecuadas para la lectura del *bitstream*. Con base en este criterio, se selecciona SPI como interfaz de comunicación para el subsistema de almacenamiento

Subistema de control

Encargado de seleccionar el modo de arranque del SoC FPGA, enviar los comandos de inicialización y coordinar el proceso de configuración del dispositivo. Para cumplir estas tareas, es importante que este subsistema cumpla con las siguientes especificaciones técnicas:

- Componente perteneciente a la categoria COTS: se prioriza la selección de un dispositivo reconfigurador perteneciente a la categoría COTS, debido a que este tipo

de componentes suelen ofrecer mayor disponibilidad comercial y, dependiendo del dispositivo, acceso a documentación y herramientas de desarrollo [75]. Este criterio se considera en concordancia con el enfoque de bajo costo adoptado para el sistema.

- El subsistema debe implementar la interfaz JTAG como interfaz principal de configuración, ya que, de acuerdo con el manual UG585, de esta forma se permite el acceso al BSR y a los mecanismos de reconfiguración del SoC FPGA [17].

- Tiempo reducido de transferencia del *bitstream*: dado que el proceso de configuración del SoC FPGA implica la transferencia del *bitstream* (figura 5.2) en el menor tiempo posible, el dispositivo reconfigurador debe contar con periféricos de comunicación que operen en el rango de decenas de MHz. De manera preferente, aproximarse a 66 MHz (figura 5.3), con el fin de reducir el tiempo del proceso de reconfiguración.

- Tecnología con resistencia a la radiación: El subsistema debe incorporar tecnología que ofrezca una mayor resistencia a los efectos de radiación. Se establece que el blindaje tolere al menos 10 krad frente a TID. Este valor se obtiene a partir del concepto margen de diseño de radiación (RDM, por sus siglas en inglés), que consiste en aplicar un factor de seguridad sobre la dosis estimada de la misión para cubrir incertidumbres del entorno. En un estudio de referencia para una misión en órbita baja heliosincrónica de 600 km, con un año de duración y con un blindaje de 2 mm de aluminio, se estimó una TID = 4 krad; al aplicar un RDM = 2, se determina que se coloquen componentes que toleren hasta 8 krad [81]. Con base en ese documento, este trabajo adopta 10 krad, donde se considera un RDM = 2, como tolerancia mínima a TID para el subsistema de control.

  De igual forma, en este trabajo se adopta como valor mínimo aceptable un LETh frente a iones pesados de $15\,\mathrm{MeV \cdot cm^2/mg}$ para dispositivos de silicio, ya que por debajo de este umbral la normativa ECSS indica que debe analizarse adicionalmente la sensibilidad inducida por protones, además de los efectos por iones pesados. En el intervalo de $15 - 60\,\mathrm{MeV \cdot cm^2/mg}$ el componente se considera dentro de un rango aceptable sujeto a análisis de SEE por iones pesados, mientras que para valores superiores a $60\,\mathrm{MeV \cdot cm^2/mg}$ se considera una condición de inmunidad práctica en el ambiente espacial, al no requerir análisis adicional según dicha normativa [82]. Por ello, valores inferiores a $15\,\mathrm{MeV \cdot cm^2/mg}$ solo se aceptan si se justifican mediante un análisis específico o si se consideran técnicas de mitigación a nivel sistema.

Con el objetivo de definir la arquitectura general del sistema de reconfiguración y establecer la separación en subsistemas, se presenta el diagrama de bloques correspondiente a esta etapa de diseño a nivel de sistema.

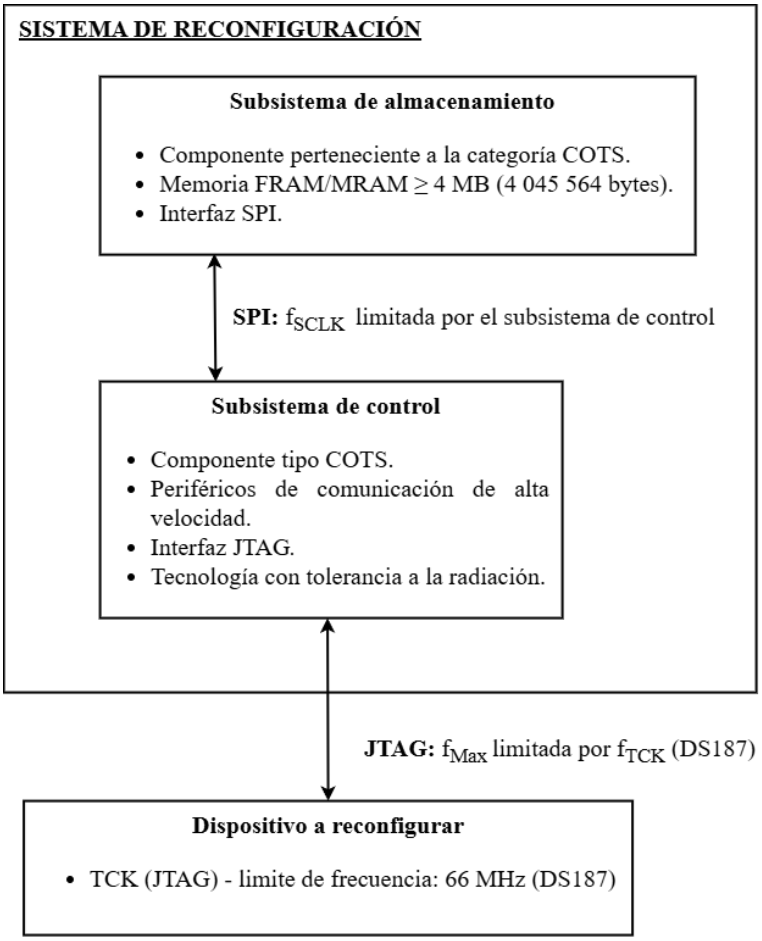


Figura 5.4: Diagrama de bloques del sistema de reconfiguración en la etapa de diseño a nivel de sistema.
Se presenta la identificación de subsistemas, la interfaz SPI y JTAG con sus respectivas restricciones de cada uno.

5.3.3. Diagrama de estados del sistema de reconfiguración

El procedimiento de reconfiguración del SoC FPGA *Xilinx Zynq* (XC7Z020) debe seguir, de manera precisa, una serie de pasos dictados en el manual UG470 [19], con el objetivo de garantizar la configuración del dispositivo. Por esta razón, definir con claridad cada etapa del procedimiento resulta de gran relevancia para estructurar adecuadamente el algoritmo y reducir la probabilidad de errores durante su implementación. De esta forma, se favorece un desarrollo más ordenado del prototipo y se disminuyen los riesgos asociados a una ejecución no planificada del proceso.

El diagrama de estados presentado en la figura 5.5 modela, de forma general, el comportamiento del sistema de reconfiguración. En él, se presentan los estados principales del sistema y los eventos que provocan la transición entre ellos. Así, se establece que la evolución del sistema no depende únicamente de cambios en señales individuales, como es el caso del diagrama de estados TAP (figura 2.2), sino a través de eventos que se presentan a lo largo del proceso de reconfiguración.

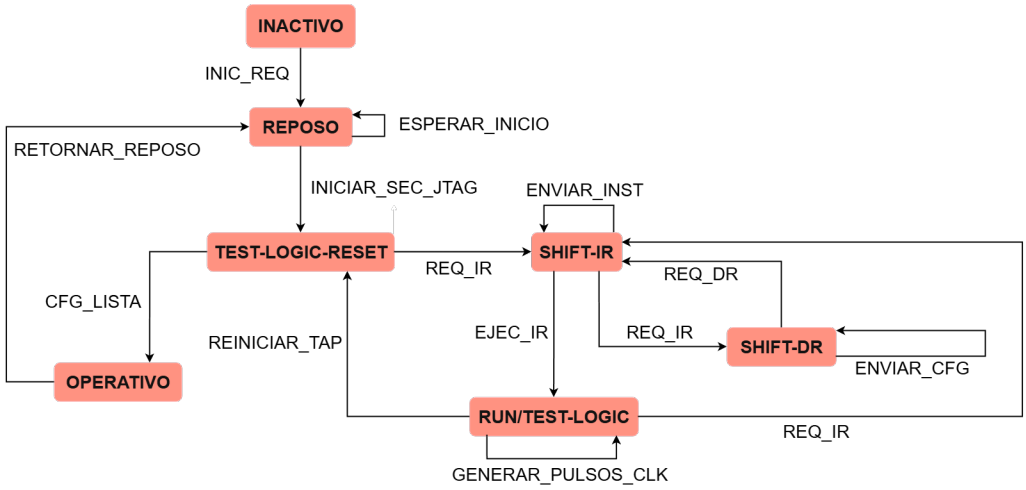


Figura 5.5: Diagrama de estados del sistema de reconfiguración.

El proceso inicia en el estado inactivo, en el cual el sistema permanece ahí hasta recibir la señal de inicialización. Posteriormente, pasa al estado de reposo, donde espera el inicio de la secuencia JTAG. Una vez iniciada, el sistema entra al estado *test-logic-reset*, en el que se reinicia el controlador TAP. A continuación, el sistema se desplaza al estado *shift-ir* para enviar las instrucciones necesarias al dispositivo. Después de cada envío de instrucción, pasa al estado *run/test-logic*, donde se generan pulsos de reloj para consumir tiempo sin transmitir información adicional. Cuando la configuración requiere el envío del *bitstream*, el sistema entra al estado *shift-dr*, en el cual se transmite la información de configuración. Una vez completado el envío de instrucciones y datos, el sistema regresa al estado *test-logic-reset*, con lo que se da por concluido el proceso de configuración. Finalmente, el sistema pasa al estado operativo y retorna a reposo, lo que lo deja disponible para un nuevo ciclo de reconfiguración.

5.4. Diseño de detalle

En este último nivel de diseño se desarrolla la implementación detallada del sistema de reconfiguración. A partir de las especificaciones técnicas definidas en la etapa anterior de diseño, se seleccionan los materiales, componentes y circuitos que conforman los subsistemas identificados previamente (subsección 5.3.2), y se diseñan los algoritmos de acuerdo con los requisitos del sistema (subsección 5.2.2). De esta manera, este apartado establece el vínculo entre las decisiones de diseño previamente definidas y la selección de componentes. Asimismo, se exponen las consideraciones de implementación derivadas de la arquitectura interna del SoC FPGA, las cuales fueron necesarias para el desarrollo del algoritmo de reconfiguración. Finalmente, se describe el funcionamiento interno del proceso de reconfiguración mediante un diagrama de flujo, en el que se integran las consideraciones necesarias para llevar a cabo el proceso de reconfiguración conforme a la secuencia establecida.

5.4.1. Diseño del subsistema de almacenamiento

El subsistema de almacenamiento tiene como función principal resguardar el *bitstream*, de modo que este pueda ser solicitado por el reconfigurador y enviado al dispositivo SoC FPGA *Xilinx Zynq*-7000 (XC7Z020) para su respectiva configuración. Para seleccionar la fuente de almacenamiento adecuada para este subsistema, se retoman de forma breve las especificaciones técnicas definidas en la etapa anterior de diseño (subsección 5.3.2):

- Preferencia por la selección de un componente perteneciente a la categoría COTS.
- Memoria no volátil con tecnología FRAM/MRAM.
- Capacidad de almacenamiento $\geq 4\,\text{MB}$.
- Interfaz de comunicación serial con frecuencia de operación limitada por la frecuencia de TCK del dispositivo (66 MHz).

Con base en las especificaciones establecidas, se prioriza la selección de componentes COTS, considerando principalmente su disponibilidad comercial y sus condiciones de adquisición. Como punto de comparación, se considera una memoria NOR *flash* CYRS17B512133AZMB, un dispositivo grado espacial endurecido a la radiación que presenta características favorables para este tipo de aplicaciones, con una tolerancia a TID aproximada de 300 krad y un LETh de superior a $80\,\text{MeV} \cdot \text{cm}^2/\text{mg}$. No obstante, su precio no se encuentra disponible públicamente y su adquisición requiere solicitar una cotización al fabricante. Este ejemplo permite evidenciar que, aunque los componentes específicamente desarrollados para ambientes de radiación pueden ofrecer prestaciones favorables frente a estos efectos, sus condiciones de adquisición pueden resultar menos accesibles para el desarrollo del sistema propuesto [83, 84].

Dentro de estas alternativas COTS, la selección se orienta hacia las memorias no volátiles basadas en tecnologías FRAM/MRAM. Esta elección considera principalmente sus mecanismos físicos de almacenamiento, basados en polarización ferroeléctrica y estados magnéticos (subsección 2.4.2), respectivamente, los cuales presentan características favorables frente a determinados efectos inducidos por radiación, particularmente asociados con SEU. A partir de estos criterios, se presentan y comparan los dispositivos candidatos con el propósito de seleccionar la alternativa que mejor satisfaga los requisitos establecidos para el subsistema de almacenamiento.

Características \ Modelos	AS303220401 08B0PSAY	CY15B116QSN 108BKXQ	EM064LXQADG 13ES2T	RAMXEEDMB85 RQ8MXPGBCE1
Tipo de memoria	STT-MRAM	FRAM	STT-MRAM	FRAM
Capacidad de almacenamiento (MB)	4	2	8	1
Frecuencia de operación máxima (MHz)	108	108	133	108
Protocolos de comunicación	SPI y QSPI	SPI, DSPI y QSPI	SPI, DSPI, QSPI y OSPI	SPI, DSPI y QSPI
Temperatura de operación (°C)	Comercial: 0 *a* 70 Industrial: −40 *a* 85 Extendido: −40 *a* 125	−40 *a* 105	Comercial: 0 *a* 70 Industrial: −40 *a* 85 Extendido: −40 *a* 105	−40 *a* 105
Voltaje de operación (V)	Núcleo: 1,71 − 2 E/S 2,70 − 3,60	Núcleo: 1,71 − 1,89 E/S 1,8 − 3,6	Núcleo: 1,65 − 2 E/S 1,65 − 2	2,7 *a* 3,6
Ciclos de escritura/borrado	10^{16}	10^{14}	Sin límite en el tiempo de vida del producto	$10^{13} a 10^{14}$
Retención de datos (años)	20	151	10	10 a 20
Tolerancia a la radiación TID (krad)	1 000 − 10 000 (tabla 2.1)	En el orden de Mrad (tabla 2.1)	> 1 000 [52, 85, 86]	En el orden de Mrad (tabla 2.1)
LETh (MeV · cm^2/mg)	≈ 69 (tabla 2.1)	Inmune (tabla 2.1)	≈ 69 [52, 85, 86]	Inmune (tabla 2.1)
Precio (USD)	Consultar con el fabricante	38,82	129,56	44,54

Tabla 5.1: Comparación de memorias no volátiles en su desempeño y tolerancia a la radiación. Los valores presentados son aproximaciones y pueden variar en función del fabricante, tecnología de fabricación y las condiciones de caracterización.
Elaboración propia con base en: [87, 88, 89, 90, 91, 92, 93]

A partir de la tabla 5.1, se observa que tanto las tecnologías FRAM como MRAM presentan características de interés para el subsistema de almacenamiento, de acuerdo con el análisis realizado previamente en la subsección 2.4.2. Sin embargo, la capacidad de almacenamiento constituye un criterio determinante para la selección, debido a que se estableció un valor de 4 MB. En la revisión de memorias FRAM COTS disponibles comercialmente no se identificaron dispositivos individuales que alcanzaran dicha capacidad. La familia EXCELON FRAM de *Infineon* dispone de dispositivos de hasta 2 MB, mientras que la línea de FRAM de *RAMXEED* alcanza actualmente 1 MB [90, 94]. Por lo tanto, las alternativas consideradas en la tabla requerirían al menos dos y cuatro circuitos integrados, respectivamente para satisfacer la capacidad mínima establecida.

La limitada capacidad de almacenamiento de las memorias FRAM dentro de la categoría COTS se encuentran relacionadas con las dificultades de escalamiento de las tecnologías ferroeléctricas convencionales basadas en titanato de circonato de plomo (PZT, por sus siglas en inglés). *RAMXEED* señala que el espesor del material ferroeléctrico empleado, cercano a 75 nm, dificulta la reducción de las dimensiones del capacitor de almacenamiento. Como referencia, con su celda PZT de producción de aproximadamente 0,5 μm^2, una matriz implementada en un chip de 5 x 5 mm^2 alcanza aproximadamente 2 MB; incluso consideraron las dimensiones mínimas teóricas propuestas para este tipo de capacitor, la capacidad estimada es de aproximadamente 5,125 MB. Debido a estas limitaciones, actualmente se investigan materiales ferroeléctricos alternativos, como el óxido de hafnio zirconio (HZO, por sus siglas en inglés), con el propósito de incrementar la capacidad de almacenamiento [95]. En consecuencia, aunque FRAM conserva las características de interés estudiadas previamente respecto a retención de datos, ciclos de escritura y comportamiento frente a determinados efectos de radiación, las alternativas COTS identificadas no permiten cumplir con el requisito de almacenamiento mediante un único dispositivo.

Por otro lado, las dos alternativas MRAM consideradas satisfacen el requisito de capacidad de almacenamiento en un solo circuito integrado. Ambos dispositivos están basados específicamente en tecnología de torque por transferencia de espín (STT, por sus siglas en inglés), una variante de las memorias MRAM que conserva el principio de almacenamiento mediante estados magnéticos en una unión de túnel magnético (MTJ, por sus siglas en inglés), pero emplea una corriente polarizada en espín para modificar la orientación magnética de la capa libre durante la operación de escritura [96]. Por lo tanto, STT-MRAM mantiene el principio de almacenamiento magnético descrito previamente para MRAM, diferenciándose principalmente en el mecanismo utilizado para realizar la conmutación del estado almacenado.

Dentro de estas alternativas, una vez analizadas sus características, se selecciona la memoria EM064LXQADG13ES2T perteneciente a *Everspin Technologies* como dispositivo de almacenamiento externo. La selección se fundamente en su capacidad de 8 MB, además de permitir satisfacer este requisito con un único circuito integrado. Asimismo, su interfaz SPI es compatible con la arquitectura de comunicación definida y el dispositivo se encuentra disponible a través de distribuidores comerciales. Sin embargo, se selecciona principalmente por su mecanismos de almacenamiento magnético, considerada en la subsección 2.4.2 por resistir más los efectos inducidos por la radiación.

5.4.2. Diseño del subsistema de control

El subsistema de control es el encargado de realizar el proceso de inicialización y configuración del SoC FPGA *Xilinx Zynq* (XC7Z020). Transfiere el *bitstream* y ejecuta comandos de control en las distintas etapas del proceso. Asimismo, este subsistema actúa como intermediario entre el subsistema de almacenamiento y el SoC FPGA.

Para diseñar este subsistema, es importante retomar las especificaciones técnicas que guían en la selección del componente a utilizar; ya que, si bien en el mercado existen múltiples dispositivos capaces de cumplir con las especificaciones, se debe elegir aquel que se adapte a las necesidades del sistema. Esta decisión no se puede tomar con base solamente en las características técnicas de una sola opción, sino que requiere de un análisis más profundo de las condiciones, restricciones y objetivos del sistema en general. A continuación, se presentan brevemente las especificaciones técnicas deducidas en la etapa anterior de diseño (subsección 5.3.2) para el subsistema de control.

- Selección preferente de un componente COTS.
- Implementación del estándar JTAG. Frecuencia máxima de operación de TCK de 66 MHz.
- Tolerancia a TID mayor o igual a 10 krad y LETh mayor o igual a 15 $\text{MeV} \cdot \text{cm}^2/\text{mg}$ reportado.

Con base en las especificaciones descritas, es importante definir la categoría de dispositivos digitales para la implementación de este subsistema. En principio, pueden considerarse distintas alternativas, como dispositivos lógicos programables o microcontroladores. En la tabla 5.2 se presenta una comparación entre estas opciones.

Tecnología	**Características principales**	**Modo de operación**	**Aplicaciones típicas**
Microcontrolador	CPU, memoria y periféricos en un solo chip, bajo consumo y programación mediante *software*.	Secuencial	Control embebido, automatización, monitoreo y comunicaciones.
PLD	Lógica programable y configurable por *hardware*, mayor flexibilidad que un circuito de función predeterminada.	Paralelo	Implementación de lógica digital, control, interfaces y procesamiento digital.

Tabla 5.2: Comparación general entre microcontroladores y dispositivos PLD en función de sus características principales, modo de operación y aplicaciones típicas. Elaboración propia con base en: [97, 98, 99]

.

Las funciones requeridas corresponden principalmente a la ejecución secuencial del proceso de inicialización, la transferencia del *bitstream* y la comunicación con el SoC FPGA mediante la interfaz JTAG. Por lo que se observa, no se requiere un procesamiento paralelo complejo, sino un dispositivo capaz de ejecutar una secuencia de control programable, establecer comunicación con otros elementos del sistema y adaptarse con facilidad a modificaciones en diferentes etapas del proyecto.

Con base en esta comparación, se prioriza el uso de un microcontrolador para la implementación del subsistema de control. Si bien un dispositivo PLD podría realizar las funciones requeridas, sus capacidades de procesamiento paralelo no representan una ventaja determinante para la arquitectura propuesta. Asimismo, el empleo de este tipo de dispositivos puede implicar mayores costos y recursos de desarrollo respecto a una solución basada en microcontroladores [100]. Considerando estos aspectos, la búsqueda del dispositivo reconfigurador se acota a la categoria de microcontroladores.

La selección del microcontrolador destinado a la implementación de este subsistema se realiza mediante el análisis comparativo de cuatro dispositivos COTS presentado en la tabla 5.3. Se prioriza que pertenezcan a la categoria COTS, puesto que los dispositivos destinados a aplicaciones espaciales o endurecidos a la radiación no son accesibles para la mayoría del público general. Como punto de referencia el microcontrolador MSP430FR5969-SP es un dispositivo de grado espacial cuyas características de tolerancia a la radiación en TID $= 50\,\text{rad}$ y LETh $= 72\,\text{MeV} \cdot \text{cm}^2/\text{mg}$ son muy convenientes para la aplicación del sistema; sin embargo, cuenta con un precio de 3669,45 USD, lo que lo deja fuera de la categoría COTS [101, 102, 103], otro ejemplo es el microcontrolador VA10800 cuyas características de tolerancia a la radiación son para TID $> 300\,\text{rad}$ y un umbral LETh prácticamente inmune, esto gracias a su tecnología *HardSil*, más que idoneas para el proyecto; sin embargo, tanto la información como el precio se tienen que solicitar al fabricante lo que lo deja fuera de adquisición fácil para el púbico general [104, 105].

Para la selección del dispositivo reconfigurador, se consideran características como la frecuencia de operación, los recursos de memoria, los módulos de comunicación, el número de líneas GPIO, el rango de temperatura de operación y el comportamiento frente a

la radiación reportado para cada dispositivo. La disponibilidad de un periférico JTAG maestro dedicado no se considera un criterio diferenciador, debido a que la interfaz JTAG será implementada mediante líneas GPIO más la interfaz SPI.

Características \ Modelos	MSP430FR5969	STM32F103RGT6	ATMEGA128-16AN	MK66FX1M0VMD18
Arquitectura	RISC de 16 bits	RISC de 32 bits	RISC de 8 bits	RISC de 32 bits
Memoria no volátil	FRAM de 64 KB	*Flash* de 1 MB	Flash de 128 KB + EEPROM de 4 KB	*Flash* de 2 MB
Memoria Volátil	SRAM de 2 KB	SRAM de 92 KB	SRAM de 4 KB	SRAM 256 KB
Frecuencia de operación (MHz)	16		16	180
Modos de bajo consumo $\left(\frac{\mu A}{MHz}\right)$	Mod. activo ≈ 100 *Standby* $\approx 0{,}4$ *Real-time clock* $\approx 0{,}25$ *Shutdown* $\approx 0{,}02$	*Sleep*, *Stop* y *Standbye*	*Idle*, *ADC Noise Reduction*, *Power-save*, *Power-down*, *Standby* y *Extended Standby*	Cuenta con múltiples modos de bajo consumo
Rango de alimentación (V)	$1{,}8 - 3{,}6$	$2{,}0 - 3{,}6$	$4{,}5 - 5{,}5$	$1{,}71 - 3{,}6$
Módulos de comunicación serial	UART x 2, SPI x 3 y I2C x 1	USART x 5, SPI x 3 y I2C x 2	USART x 2, SPI x 1 y TWC/I2C x 1	SPI x 3, I2C x 4 y UART x 5
Rango de temperatura de operación (°C)	-40 *a* 85	-40 *a* 105	-40 *a* 105	-40 *a* 105
Tolerancia a TID (krad)	Modo activo: > 490 Modo *sleep*: $\approx 7{,}5$ [106]	25,1 [107]	$\approx$ 30 [108]	$\approx$ 35 [109]
LETh reportado ($MeV \cdot cm^2/mg$)	0,3 (SEU) [110]	62,5 (SEL) [111]	4,5 (SEU) [112]	No identificado
Precio (USD)	15,99	20,05	20,50	39,62

Tabla 5.3: Análisis comparativo de alternativas de microcontroladores. Los valores presentados son aproximaciones y pueden variar en función del fabricante, tecnología de fabricación y otros factores.
Elaboración propia con base en: [101, 113, 106, 110, 114, 115, 111, 107, 116, 117, 108, 112, 118, 119, 109]

A partir de la comparación presentada en la tabla 5.3, se observa que los cuatro modelos cuentan con recursos suficientes para implementar las funciones principales del subsistema de control. Sin embargo, algunas alternativas ofrecen capacidades de procesamiento superiores a las requeridas para esta aplicación, mientras que otras presentan menores recursos de comunicación o cuentan con información limitada respecto a su utilización previa en aplicaciones espaciales. Por esta razón, la selección no se realiza únicamente con base en la capacidad de procesamiento, sino considerando también el tipo de memoria, consumo de potencia, disponibilidad, costo y antecedentes de utilización en sistemas espaciales.

Bajo estos criterios, se selecciona el MSP430FR5969. El reporte *State of the Art Small Spacecraft Technology* de la NASA, incluye dentro de los sistemas de cómputo a bordo al *Eddi OBC* de *Spacemanic* basado en un microcontrolador de la familia MSP430, para el cual se reporta herencia de vuelo en órbita baja terrestre [120]. Este antescedente muestra la utilización de dispositivos pertenecientes a esta arquitectura en sistemas de cómputo destinados a pequeñas naves espaciales. De manera más específica, el *CubeSat* MiTEE-1 empleó cuatro microcontroladores MSP430FR5969 como parte de su sistema de control y manejo de datos, y fue lanzado exitosamente en 2021 [121]. Este antecedente no implica que el dispositivo COTS se encuentre calificado para aplicaciones espaciales; sin embargo, proporciona evidencia de su utilización previa en *hardware* de vuelo y constituye un criterio favorable para su selección.

Por lo anterior, el MSP430FR5969 se considera la alternativa más conveniente para la implementación del subsistema de control, al proporcionar los recursos funcionales requeridos, características de bajo consumo y memoria FRAM cuya resistencia a la radiación es

de suma importancia para el sistema, además de contar con antecedentes documentados de utilización en una misión *CubeSat*.

5.4.3. Diagrama de bloques del sistema de reconfiguración

Con base en las decisiones de diseño adoptadas para los subsistemas de almacenamiento y control, se presenta el diagrama de bloques correspondiente en la figura 5.6. Esta representación da a conocer la integración de los componentes principales de cada subsistema y las interfaces de comunicación que se emplean para lograr exitosamente la reconfiguración del SoC FPFA.

De esta forma, el diagrama ofrece una visión completa de la arquitectura implementada, lo que permite identificar la relación entre los elementos seleccionados y la forma en que interactúan dentro de la solución. Con ello, se resume de manera gráfica la estructura final del sistema, resultado del proceso de diseño de detalle.

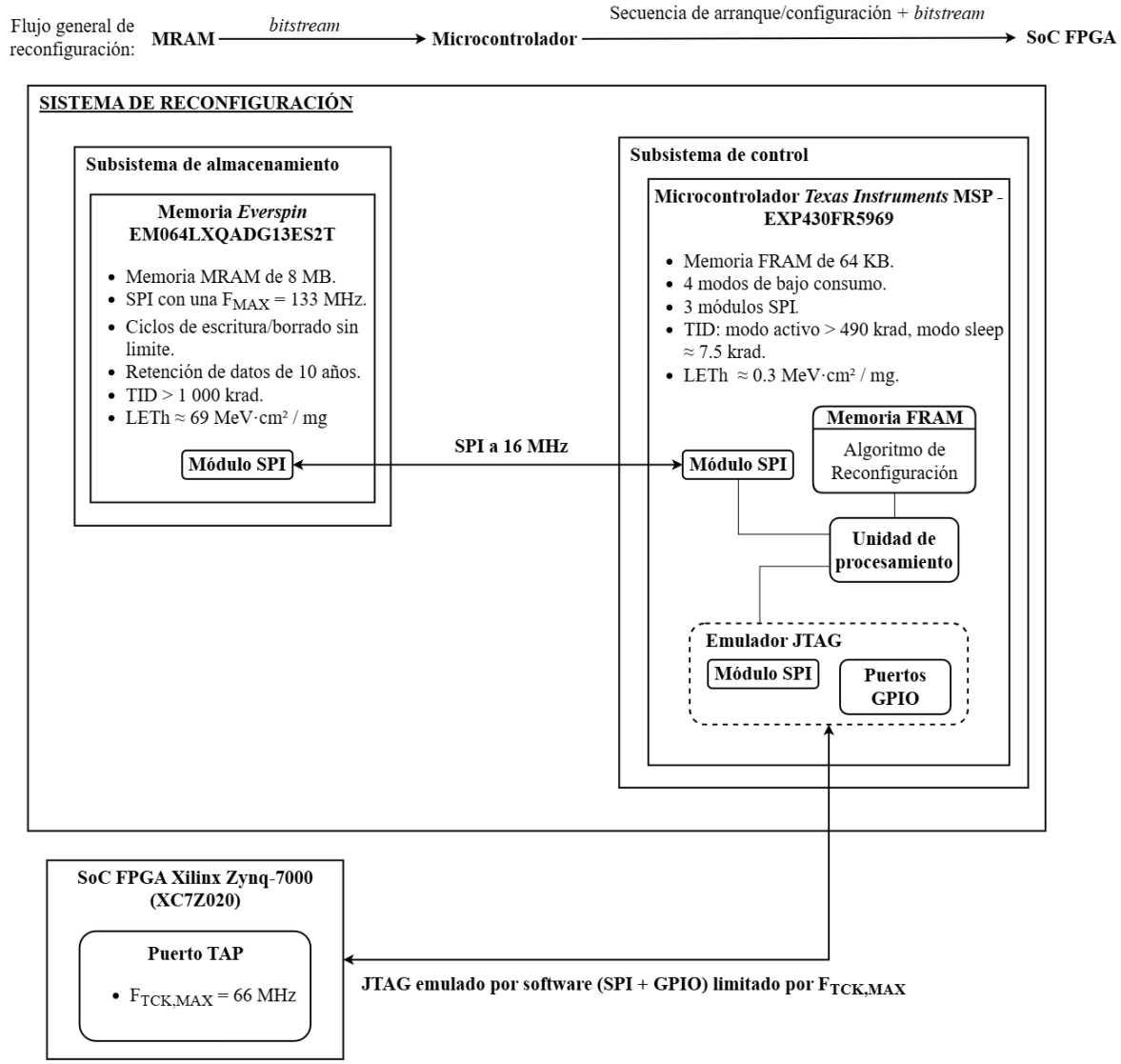


Figura 5.6: Diagrama de bloques del sistema de reconfiguración.
Etapa de diseño de detalle.

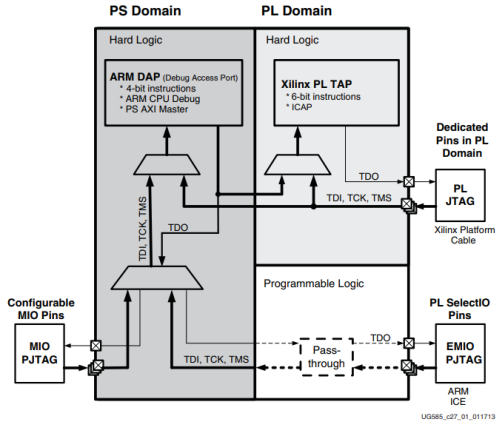
5.4.4. Consideraciones arquitectónicas del SoC FPGA

En este apartado se presentan las consideraciones arquitectónicas del dispositivo a reconfigurar; es decir, las particularidades de la arquitectura interna del SoC FPGA tomadas en cuenta para el desarrollo del sistema. Para comprender dichas consideraciones, es importante distinguir entre un FPGA convencional y un SoC FPGA.

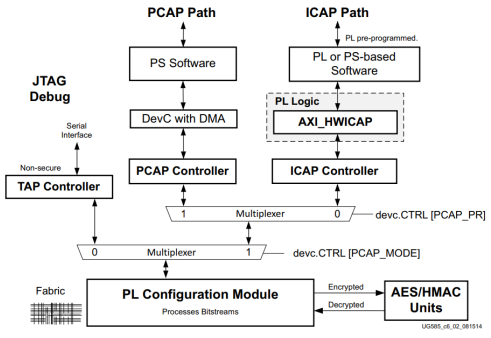
Un FPGA es un dispositivo reconfigurable que permite implementar funciones digitales mediante lógica programable, en la cual se integra la funcionalidad del sistema. Dicha lógica puede configurarse a través de un archivo de configuración, conocido como *bitstream* (sección 2.1) para satisfacer necesidades específicas [122].

Por otro lado, un SoC FPGA es, en términos generales, un dispositivo que integra en un mismo chip un sistema de procesamiento, basado en uno o más procesadores, y una región de lógica programable. Esta arquitectura permite combinar la ejecución de *software* con la implementación de *hardware* reconfigurable [123], lo que proporciona mayor flexibilidad y capacidad de procesamiento en comparación con un FPGA.

De acuerdo con el manual UG585 [17], la arquitectura interna del SoC FPGA *Xilinx Zynq*-7000 (XC7Z020) se organiza en dos secciones principales: la lógica programable (PL) y el sistema de procesamiento (PS). Cada una de estas secciones dispone de controladores y recursos propios de configuración, lo que define diferentes funciones dentro del dispositivo, como se muestra en la figura 5.7a. Asimismo, se identifican tres rutas principales mediante las cuales pueden llevarse a cabo la programación del sistema, ya sea desde la región PL o desde la región PS, donde se emplean distintos controladores internos, como se muestra en la figura 5.7b.



(a) División general de la arquitectura interna del SoC FPGA en los dominios de sistema de procesamiento (PS) y lógica programable (PL).



(b) Rutas de configuración disponibles para el SoC FPGA.

Figura 5.7: Elementos relevantes de la arquitectura interna del SoC FPGA. Fuente: [17].

Como se observa en la figura 5.8, cada uno de los dominios cuenta con su propio controlador de acceso: el controlador puerto de acceso de depuración (DAP, por sus siglas en inglés) para la región PS y el controlador TAP para la región PL. Por configuración de fabrica, ambos controladores se encuentran conectados en serie, en una configuración cascada (*cascade*). No obstante, mediante la modificación de ciertos registros internos, esta configuración puede cambiarse a uno dividido, en el que cada controlador opera de manera independiente (*independent*).

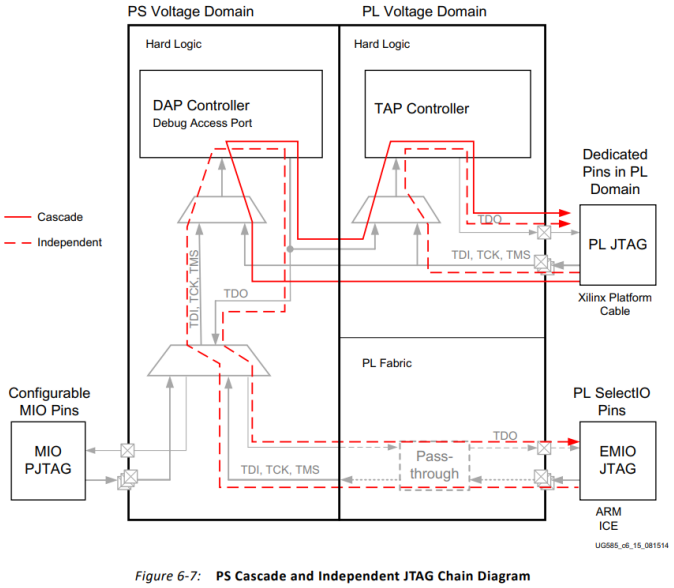


Figura 5.8: División de la arquitectura interna del SoC FPGA con sus respectivos controladores
Fuente: [17].

Esta característica resulta relevante para el desarrollo del algoritmo, ya que la forma en que el microcontrolador externo configura al dispositivo depende directamente del modo en que se encuentren conectados los controladores internos de cada dominio. Cada controlador dispone de sus propios registros de instrucciones y datos; sin embargo, cuando operan en una configuración cascada (figura 5.8), dichos registros se conectan en serie, lo que forma una sola cadena de desplazamiento. Por ello, durante el acceso al dispositivo se tienen que considerar las instrucciones que soporta cada controlador, como se muestra en la figura 5.9, y los registros de los controladores que se encuentran en la misma cadena. En el caso del registro de instrucciones, los registros de los controladores DAP y TAP se concatenan para formar un único registro de mayor longitud. De la misma forma, durante el acceso a los registros de datos, los registros correspondientes de ambos controladores también se conectan en serie, lo que forma una única cadena de desplazamiento. Por ello, el algoritmo debe contemplar esta organización en cascada para asegurar que tanto las instrucciones como los datos alcancen el controlador requerido.

IR Instruction	Binary Code[3:0]	DR Width	Description
ABORT	1000	35	JTAG-DP abort register
DPACC	1010	35	JTAG DP access register
APACC	1011	35	JTAG-AP access Register
ARM_IDCODE	1110	32	IDCODE for Arm DAP IP
BYPASS	1111	1	

(a) Instrucciones soportadas por el controlador DAP en el dominio PS.

Boundary Scan Command	Binary Code[5:0]	Description
EXTEST	000000	Enables boundary-scan EXTEST operation
SAMPLE	000001	Enables boundary-scan SAMPLE operation
USER1	000010	Access user-defined register 1
USER2	000011	Access user-defined register 2
USER3	100010	Access user-defined register 3

(b) Algunas instrucciones soportadas por el controlador TAP en el dominio PL, conforme a la referencia incluida en el manual UG585.

Figura 5.9: Instrucciones empleadas para el acceso y control de las regiones PS y PL a través de sus respectivos controladores internos.
Fuente: [17].

5.4.5. Análisis del *bitstream* y herramienta de decodificación

Como se planteó en la etapa de diseño conceptual (subsección 5.2), se requiere analizar con mayor profundidad la estructura, organización y orden de transmisión del *bitstream*. Para ello, se desarrolla una herramienta de decodificación que permite analizar su contenido a nivel binario e identificar los distintos elementos que intervienen durante el proceso de configuración. Este análisis sirve como apoyo para comprender la organización de la información que posteriormente será transmitida hacia la lógica programable del SoC FPGA.

La herramienta de decodificación no forma parte del sistema de reconfiguración, sino que se emplea como un recurso de apoyo durante la etapa de diseño para facilitar el desarrollo del algoritmo de reconfiguración y análisis del *bitstream*. Su desarrollo se basa en el manual UG470 [19], en el cual se describe la estructura del *bitstream*, así como los distintos tipos de paquetes, encabezados, comandos y elementos necesario para la configuración de dispositivos FPGA pertenecientes a la serie 7 de *Xilinx*. Esta misma referencia se toma como base para el dominio PL del SoC FPGA, puesto que también pertenece a dicha serie.

El *bitstream* se encuentra organizado como una secuencia de palabras de 32 bits que cumplen con distintas funciones dentro del proceso de configuración. Antes de la información correspondiente a los paquetes de configuración, pueden encontrarse palabras de relleno (*dummy words*), patrones empleados para la detección del ancho del bus (figura 5.10) y palabras sin operaciónes (*NOP*). Dentro de esta secuencia destaca la palabra de sincronización 0xAA995566 (figura 5.11), cuya función es indicar a la lógica de configuración el inicio de la información válida y establecer la alineación necesaria para interpreta las palabras posteriores.

Table 5-2: **Bus Width Detection Pattern**

D[24:31]	D[16:23]	D[8:15]	D[0:7]	Comments
0xFF	0xFF	0xFF	0xFF	
0x00	0x00	0x00	0xBB	Bus Width Pattern
0x11	0x22	0x00	0x44	Bus Width Pattern
0xFF	0xFF	0xFF	0xFF	
0xFF	0xFF	0xFF	0xFF	
0xAA	0x99	0x55	0x66	Sync Word
...	...	...	...	...

Figura 5.10: Patrones de detección del ancho del bus.
Fuente: [19].

Table 5-3: **Sync Word**

31:24	23:16	15:8	7:0
0xAA	0x99	0x55	0x66

Figura 5.11: Palabra de sincronización perteneciente a dispositivos de la serie 7 de *Xilinx*.
Fuente: [19].

Una vez identificada la palabra de sincronización, la lógica de configuración interpreta las palabras posteriores principalmente mediante dos tipos de paquetes: tipo 1 y tipo 2. Los paquetes tipo 1 se utilizan para realizar operaciones sobre los registros de configuración del dispositivo. Su encabezado contiene información que permite identificar el tipo de paquete, la operación que debe realizarse, el registro involucrado y la cantidad de palabras de datos asociadas. De esta manera, a partir de una sola palabra es posible determinar si la operación corresponde, por ejemplo, a una escritura sobre un registro o al acceso a algún registro relacionado con el proceso de configuración (figura 5.12).

Table 5-20: **Type 1 Packet Header Format**

Header Type	Opcode	Register Address	Reserved	Word Count
[31:29]	[28:27]	[26:13]	[12:11]	[10:0]
001	xx	RRRRRRRRRxxxxx	RR	xxxxxxxxxxx

Notes:

1. "R" means the bit is not used and reserved for future use. The reserved bits should be written as 0s.

Table 5-21: **OPCODE Format**

OPCODE	Function
00	NOP
01	Read
10	Write
11	Reserved

Figura 5.12: Paquetes tipo uno.
Fuente: [19].

Los paquetes tipo 2 se emplean cuando es necesario transferir información. A diferencia de los paquetes tipo 1, su encabezado no contiene una nueva dirección de registro, sino que utiliza la establecida previamente mediante un paquete tipo 1. Por esta razón, ambos tipos de paquetes se encuentran relacionados: un paquete tipo 1 establece inicialmente la operación, el registro de destino y la cantidad de información a transferir, mientras que un paquete tipo 2 permite continuar la transferencia de información asociado con dicha operación (figura 5.13). Esta estructura resulta particularmente relevante durante la carga de los datos que representan la configuración de la lógica programable.

Table 5-22: **Type 2 Packet Header**

Header Type	Opcode	Word Count
[31:29]	[28:27]	[26:0]
010	XX	xxxxxxxxxxxxxxxxxxxxxxxxxxx

Figura 5.13: Paquetes tipo dos.
Fuente: [19].

Además de los paquetes mencionados, dentro del *bitstream* aparecen comandos y accesos a distintos registros internos que controlan etapas específicas de la configuración (figura 5.14). De esta forma, el archivo no constituye únicamente un bloque continuo de datos correspondientes al diseño implementado, sino a una secuencia estructurada que

combina palabras de sincronización, encabezados de paquetes, operaciones sobre registros, comandos de configuración y bloques de datos. La correcta identificación de estos elementos permite establecer donde inicia cada sección y cuál es la función que desempeña dentro de la secuencia.

Table 5-23: **Type 1 Packet Registers**

Name	Read/Write	Address	Description
CRC	Read/Write	00000	CRC Register
FAR	Read/Write	00001	Frame Address Register
FDRI	Write	00010	Frame Data Register, Input Register (write configuration data)
FDRO	Read	00011	Frame Data Register, Output Register (read configuration data)
CMD	Read/Write	00100	Command Register
CTL0	Read/Write	00101	Control Register 0
MASK	Read/Write	00110	Masking Register for CTL0 and CTL1
STAT	Read	00111	Status Register

Figura 5.14: Algunos de los registros de los paquetes tipo uno.
Fuente: [19].

La figura 5.15b presenta algunos de los elementos descritos anteriormente y su representación dentro del archivo RBT. El análisis de esta estructura constituye la base para evaluar posteriormente el funcionamiento de la herramienta de decodificación y su capacidad de reconocer correctamente los elementos presentes en los archivos de configuración utilizados durante las pruebas.

Configuration Data Word (hex)	Description
000000BB	Bus width auto detect, word 1
11220044	Bus width auto detect, word 2
FFFFFFFF	Dummy pad word
FFFFFFFF	Dummy pad word
AA995566	Sync word
20000000	NOOP
30022001	Packet Type 1: Write TIMER register, WORD_COUNT=1
00000000	TIMER[31:0]=00000000 (hex) (Placeholder for optional watchdog timer)
30020001	Packet Type 1: Write WBSTAR register, WORD_COUNT=1

(a) Ejemplos de palabras y comandos que conforman la estructura del *bitstream*. Incluye encabezados, palabras de sincronización, registros de control, entre otros elementos.

```
Xilinx ASCII Bitstream
Created by Bitstream 2024.1 SW Build 5076996 on Wed May 22 18:37:14 MDT 2024
Design name:    blinky;COMPRESS=TRUE;UserID=0XFFFFFFFF;Version=2024.1
Architecture:   zynq
Part:           7z010clg400
Date:           Wed Oct 01 09:23:05 2025
Bits:           1732480
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
00000000000000000000000010111011
00010001001000100000000001000100
```

(b) Representación del archivo *bitstream* en formato de texto, donde se observa la secuencia de datos utilizada durante el proceso de configuración.

Figura 5.15: Ejemplo de los elementos que conforman la estructura del archivo *bitstream* y su representación en formato de texto.
Fuente: [19].

5.4.6. Diagrama de flujo del proceso de reconfiguración

El diagrama de flujo 5.16 representa los pasos que se implementan en el algoritmo para realizar la reconfiguración del SoC FPGA. Asimismo, presenta consideraciones dentro del proceso de reconfiguración, resultado de la arquitectura interna del SoC FPGA (subsección 5.4.4), lo que añade dificultad al implementar el procedimiento descrito en el manual UG470 [19], un manual exclusivo para los FPGA pertenecientes a la serie 7 de *Xilinx*.

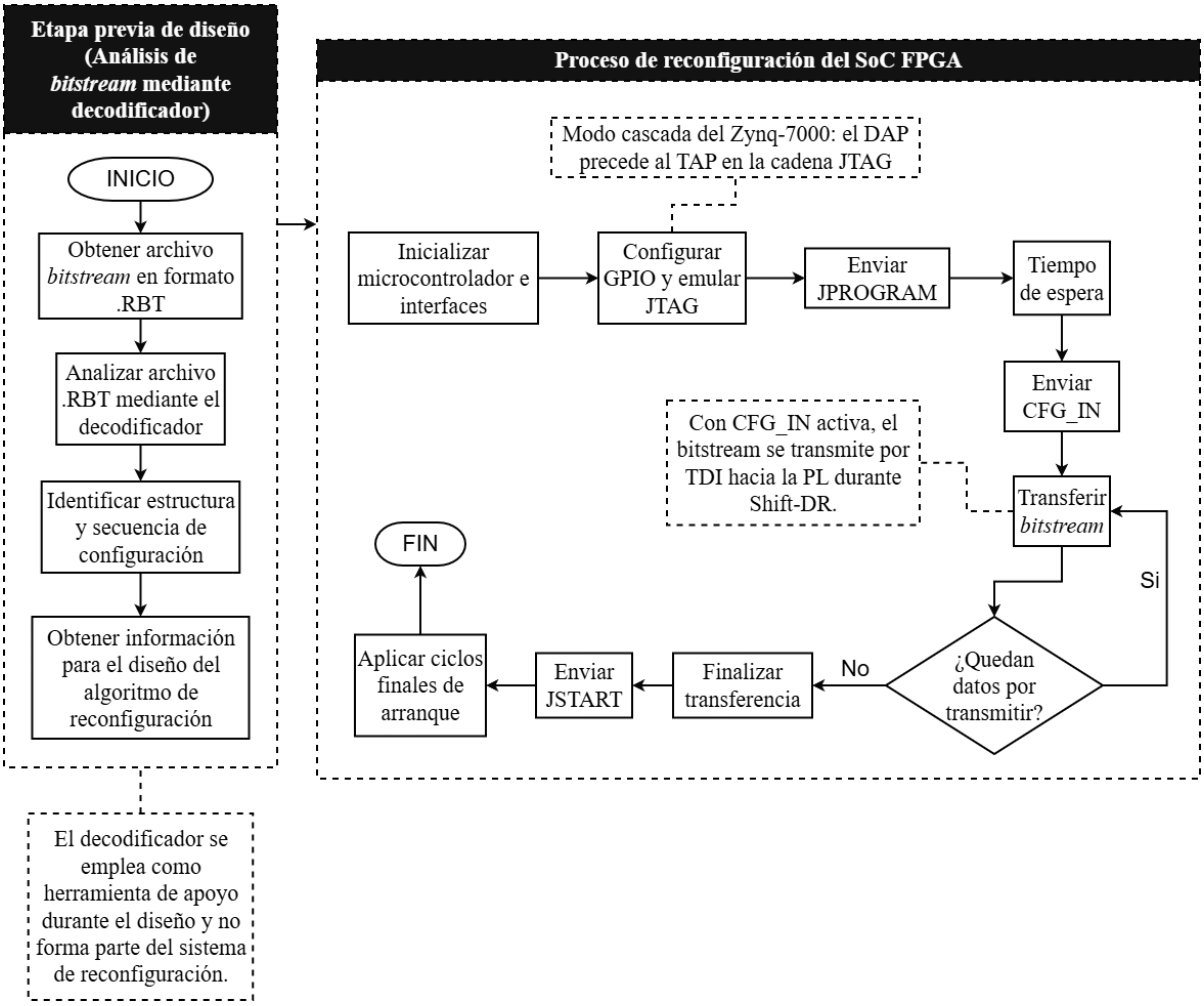


Figura 5.16: Diagrama de flujo del procedimiento de reconfiguración del SoC FPGA.

Como se observa en la figura 5.16, el desarrollo del algoritmo parte de una etapa previa de análisis del archivo *bitstream*. Para ello, se emplea la herramienta de decodificación descrita en la subsección 5.2.3, mediante la cual se identifica la estructura del archivo de configuración y la secuencia de los elementos que lo conforman. La información obtenida se utiliza como apoyo para establecer el manejo del *bitstream* dentro del procedimiento de reconfiguración; sin embargo, el decodificador no interviene durante la operación del sistema final.

Una vez establecida la información necesaria para el proceso de configuración, el algoritmo inicia con la preparación del microcontrolador y de las interfaces empleadas para comunicarse con el SoC FPGA. La emulación de JTAG se realiza mediante el control

de señales correspondientes desde el MSP430FR5969. En esta etapa debe considerarse la arquitectura interna del Zynq-7000 descrita en la subsección 5.4.4, debido a que, en el modo de operación empleado, el DAP correspondiente al PS y el TAP asociado con la PL forman parte de la misma cadena JTAG. Por esta razón, las instrucciones dirigidas a la lógica programable no pueden enviarse considerando únicamente al TAP, sino que deben construirse teniendo en cuenta la presencia de ambos controladores.

La primera instrucción utilizada en el proceso es JPROGRAM, con la cual se borra la configuración previamente cargada en la memoria de configuración interna. Después de su envío, el algoritmo lleva el controlador TAP al estado *run-test/idle*, donde permanece durante el intervalo requerido para completar la inicialización de la lógica de configuración antes de continuar con la cargar del archivo. Posteriormente, se envía la instrucción CFG_IN, la cual selecciona la ruta utilizada para introducir los datos de configuración a través de JTAG. A partir de este punto, el controlador TAP es llevado al estado *shift-dr*, desde donde se realiza el desplazamiento del *bitstream* hacia la lógica de configuración.

Para implementar esta etapa se emplean de manera complementaria los periféricos GPIO y SPI del microcontrolador. Las transiciones entre los estados del controlador TAP y el envío controlado de las instrucciones se realizan mediante GPIO, lo que permite establecer de manera explícita los valores de TMS, TDI y TCK requeridos por la secuencia JTAG. Por otro lado, durante la transferencia continua del *bitstream*, el módulo SPI se utiliza para efectuar el envío de la información binaria. Esta combinación permite conservar el control requerido para la navegación del controlador TAP y, al mismo tiempo, aprovechar el periférico serial para la transferencia del archivo de configuración.

El algoritmo mantiene la transferencia mientras existan datos pendientes del *bitstream*. Para ello, el microcontrolador obtiene de forma secuencial la información almacenada en la memoria y controla la cantidad de datos enviados hacia el SoC FPGA. Una vez que se completa la transferencia, se abandona la etapa de desplazamiento de datos y se recupera el control mediante GPIO para continuar con la secuencia de estados JTAG necesaria para finalizar el procedimiento.

Finalmente, se envía la instrucción JSTART, utilizada para iniciar la secuencia de arranque de la lógica programable después de recibir la nueva configuración. Posteriormente, se generan los ciclos de reloj requeridos para completar esta etapa y finalizar el procedimiento de reconfiguración. De esta manera, el algoritmo integra el análisis previo del archivo de configuración, la adaptación de la secuencia JTAG a la arquitectura del SoC FGPA y la transferencia controlada del *bitstream* dentro de un mismo procedimiento.

Capítulo 6

Pruebas y resultados

En este capítulo se presentan los resultados de la implementación de los objetivos específicos. Se documentan las pruebas realizadas y los resultados recabados del desarrollo del sistema.

En primera lugar, se presentan los resultados del decodificador de *bitstream*. Su propósito es dar a conocer la estructura y la información contenida en el archivo de configuración. Estos resultados permiten observar el correcto funcionamiento del decodificador y su capacidad para reconocer los distintos elementos que conforman los archivos *bitstream*, correspondientes a la parte PL de cada dispositivo FPGA o SoC FPGA pertenecientes a la serie 7 de *Xilinx*.

Posteriormente, se presentan los resultados relacionados con el algoritmo desarrollado para entablar una comunicación entre el microcontrolador y el dispositivo SoC FPGA, por medio de la emulación del estándar JTAG. Para esta prueba, se documenta la generación de las tramas necesarias para la obtención del *IDCODE* del dispositivo SoC FPGA.

Por último, se muestran los resultados correspondientes a la transmisión del archivo *bitstream* desde una memoria hasta el dispositivo SoC FPGA. El microcontrolador funge como un intermediario durante el desarrollo de múltiples pruebas para la correcta configuración del dispositivo de interés. Asimismo, se muestran los resultados asociados a la implementación de comandos para llevar a cabo diferentes tareas de reconfiguración.

6.1. Decodificador *bitstream*

La herramienta de diseño de *Xilinx*, *Vivado*, permite generar cinco formatos distintos para la exportación del archivo de salida *bitstream*. Cada uno de ellos cuenta con características únicas que los vuelven ideales para adaptarse a diferentes propósitos. En este trabajo se empleó el formato RBT por sus características antes mencionadas (subsección 2.1.2).

```
Xilinx ASCII Bitstream
Created by Bitstream 2024.1 SW Build 5076996 on Wed May 22 18:37:14 MDT 2024
Design name:    blinky;COMPRESS=TRUE;UserID=0XFFFFFFFF;Version=2024.1
Architecture:   zynq
Part:           7z010clg400
Date:           Wed Oct 01 09:23:05 2025
Bits:           1732480
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
00000000000000000000000010111011
00010001001000100000000001000100
```

Figura 6.1: Archivo *bitstream* de un proyecto en *Vivado*.

En la figura 6.1 se muestra un fragmento del *bitstream* correspondiente a un proyecto de *Vivado*, donde se observa la información generada por la herramienta de diseño.

Para interpretar la información contenida en el archivo *bitstream* se emplea el lenguaje de programación Python, dada su facilidad para la manipulación de archivos de texto y su versatilidad en el procesamiento de datos. Se desarrolla una función que devuelve una coincidencia encontrada en el diccionario proporcionado como parámetro, de lo contrario arroja "Desconocido", como se muestra en la figura 6.2. Los parámetros con los que trabaja dicha función son:

- bit: variable que representa un bit individual o un grupo de bits que son evaluados.
- mapa: nombre del diccionario donde tiene que buscar alguna coincidencia.

```
38
39  # Funcion para decodificar bits especificos
40  def decodificador_bit(bit, mapa, default = 'Desconocido'):
41      return mapa.get(bit, default)
42
43
```

Figura 6.2: Función de búsqueda.

El *script* está diseñado para filtrar las líneas que están compuestas únicamente por unos y ceros. Posteriormente, identifica las palabras *dummy* y *nop* que pueda contener el archivo. Una vez identificadas, se procede al reconocimiento del parámetro ancho del bus, la palabra de sincronización, los paquetes tipo 1, así como la identificación del registro al que hace referencia y de la operación que se va a ejecutar. La palabra inmediata posterior al paquete tipo 1 es aquella que contiene información detallada sobre la configuración del registro previamente identificado. Es aquí donde la función 6.2 toma más relevancia puesto que cada grupo de bits tiene información única en cada registro, por lo que, uno de los parámetros que se otorga es una variable tipo diccionario que contiene todos los datos

obtenidos del manual UG470 con los cuales tiene que buscar una coincidencia. Por último, se identifican los paquetes tipo 2 y todas las palabras de configuración del *bitstream* (figura 6.3).

```
elif Register_Address_dec == 'STAT':
    Bit2625_BUS_WIDTH       = decodificador_bit(dato[5:7],{'00':'Ancho de bus = x1','01':'Ancho de bus = x8','10':'Ancho de bus = x16','11':'Ancho de bus = x32'})
    Bit2018_STARTUP_STATE   = decodificador_bit(dato[11:14],{'000':'Fase 0','001':'Fase 1','011':'Fase 2','010':'Fase 3','110':'Fase 4','111':'Fase 5','101':'Fase 6','100':'Fase 7'})
    Bit17_XADC_OVER_TEMP    = decodificador_bit(dato[14],{'0':'Sin alarma de sobretemperatura del XADC (default)','1':'Alarma de sobretemperatura del XADC'})
    Bit16_DEC_ERROR         = decodificador_bit(dato[15],{'0':'No DEC_ERROR','1':'DEC_ERROR'})
    Bit15_ID_ERROR          = decodificador_bit(dato[16],{'0':'No ID_ERROR','1':'ID_ERROR'})
    Bit14_DONE              = 'Valor del pin DONE'
    Bit13_RELEASE_DONE      = decodificador_bit(dato[18],{'0':'Señal DONE no liberada','1':'Señal DONE liberada'})
    Bit12_INIT_B            = 'Valor en el pin INIT_B'
    Bit11_INIT_COMPLETE     = decodificador_bit(dato[20],{'0':'La inicialización no ha finalizado','1':'La inicialización ha finalizado'})
    Bit108_MODE             = 'Estado de los pines de Modo'
    Bit7_GHIGH_B            = decodificador_bit(dato[24],{'0':'GHIGH_B activado','1':'GHIGH_B desactivado'})
    Bit6_GWE                = decodificador_bit(dato[25],{'0':'Los FFs y Bloques RAM están deshabilitados para escritura','1':'Los FFs y Bloques RAM están habilitados para escritura'})
    Bit5_GTS_CFG_B          = decodificador_bit(dato[26],{'0':'Todas las E/S de usuario se colocan en alta impedancia','1':'Todas las E/S se comportan según lo configurado'})
    Bit4_EOS                = decodificador_bit(dato[27],{'0':'La secuencia de arranque no ha finalizado','1':'La secuencia de arranque ha finalizado'})
    Bit3_DCI_MATCH          = decodificador_bit(dato[28],{'0':'DCI no coincidente','1':'DCI coincide'})
    Bit2_MMCM_LOCK          = decodificador_bit(dato[29],{'0':'Los MMCM no están bloqueados','1':'Los MMCM están bloqueados'})
    Bit1_PART_SECURED       = decodificador_bit(dato[30],{'0':'Seguridad del descifrador no configurada','1':'Seguridad del descifrador configurada'})
    Bit0_CRC_ERROR          = decodificador_bit(dato[31],{'0':'No CRC error','1':'CRC error'})

    archivo_destino.write(f'Bit 26:25: {Bit2625_BUS_WIDTH} | Bit 20:18: {Bit2018_STARTUP_STATE} | Bit 17: {Bit17_XADC_OVER_TEMP} | Bit 16: {Bit16_DEC_ERROR} | Bit 15: {Bit15_ID_ERROR}
```

Figura 6.3: Diccionario que permite identificar qué acciones se realizan de acuerdo con el registro identificado previamente.

Las pruebas se realizaron con varios archivos *bitstream* pertenecientes a la serie 7 (figura 6.1), con el objetivo de corroborar el funcionamiento del decodificador.

En la figura 6.4 se observa la identificación de distintos elementos que conforman la estructura del *bitstream*, entre ellos palabras *dummy*, palabras *nop*, el patrón de detección automática del ancho del bus y la palabra de sincronización. Asimismo, el decodificador reconoce los paquetes tipo 1 e interpreta los campos asociados a cada uno, como la operación a realizar, el registro de destino y el número de palabras de datos vinculadas con la operación. Estos resultados permiten comprobar que la herramienta identifica correctamente los principales elementos de control presentes en la secuencia de configuración, con respecto a lo documentado en el manual UG470 de *Xilinx* (subsección 5.4.5).

```
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
00000000000000000000000010111011: Bus Width Autodetection (0xBB, First Word)
00010001001000100000000001000100: El ancho del bus es de 8 bits (Byte mas significativo = 0x11)
11111111111111111111111111111111: DUMMY
11111111111111111111111111111111: DUMMY
10101010100110010101010110011O: Palabra de Sincronizacion. Nada se envia hasta encontrar esta palabra
00100000000000000000000000000000: NOP
00110000000000010001000000000001: Type 1 | Opcode: WRITE | Reg: TIMER | Word Count: 1
00000000000000000000000000000000: Bit 31: Watchdog deshabilitado en modo usuario (default) | Bit 30: Watchdog deshabilitado durante la
configuracion (default) | Bit 29:0: Tiempo de espera binario = 00000000000000000000000000000000, valor decimal = 0
00110000000000010000000000000001: Type 1 | Opcode: WRITE | Reg: WBSTAR | Word Count: 1
00000000000000000000000000000000: Bit 31:30: Valor por defecto para RS[1:0] en el siguiente reinicio (00) | Bit 29: Pines RS[1:0] en 3-
state habilitado (default) | Bit 28:0: Direccion de inicio = 00000000000000000000000000000000
00110000000000001000000000000001: Type 1 | Opcode: WRITE | Reg: CMD | Word Count: 1
00000000000000000000000000000000: Valor[4:0] = 00000 | Proposito de NULL: No hace nada.
00100000000000000000000000000000: NOP
00110000000000001000000000000001: Type 1 | Opcode: WRITE | Reg: CMD | Word Count: 1
00000000000000000000000000000111: Valor[4:0] = 00111 | Proposito de RCRC: Reinicia el registro CRC.
00100000000000000000000000000000: NOP
00100000000000000000000000000000: NOP
00110000000000010011000000000001: Type 1 | Opcode: WRITE | Reg: RBCRC_SW | Word Count: 1
```

Figura 6.4: Archivo *bitstream* del *blink* decodificado.

En la figura 6.5, se observa la identificación de un paquete tipo 2 con operación de escritura y un conteo de 659251 palabras. Este tipo de paquete se emplea para transferir bloques extenos de información y utiliza la dirección del registro establecida previamente por un paquete tipo 1. A continuación del encabezado, el decodificador reconoce las palabras correspondientes a los datos de configuración del *bitstream*, lo que permite verificar su capacidad para interpretar tanto los elementos de control como los bloques de información que conforman el archivo de configuración.

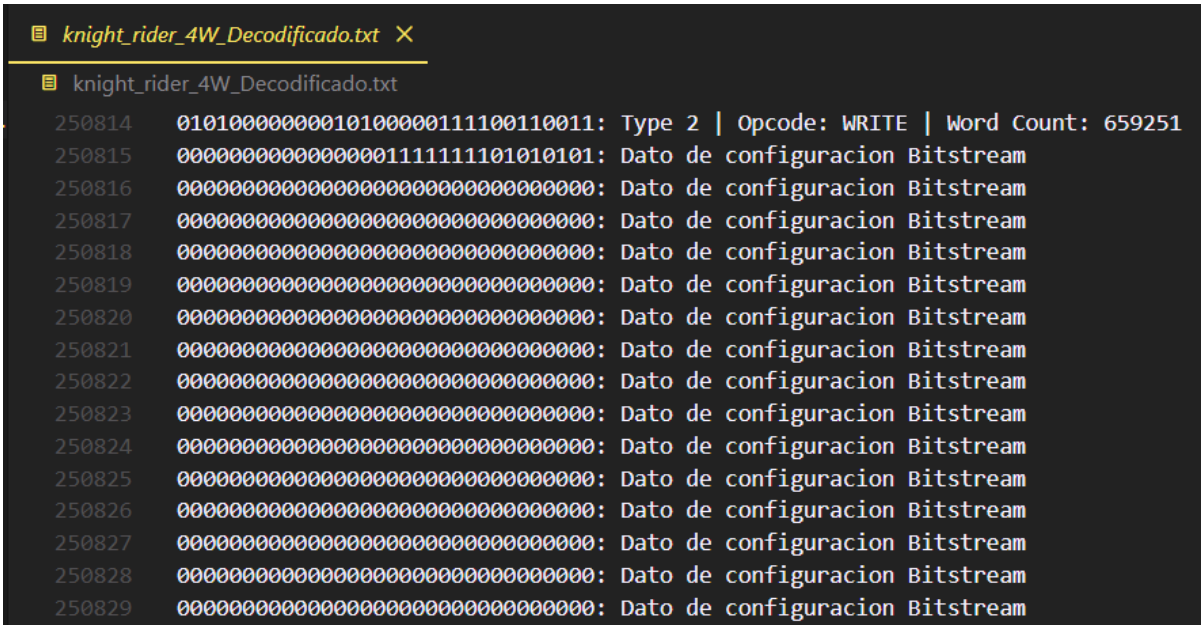


Figura 6.5: Archivo *bitstream* del *blink* decodificado.

6.2. Emulación de JTAG

Una vez analizada la estructura del *bitstream*, se implementó el algoritmo encargado de emular la interfaz JTAG mediante el microcontrolador MSP430FR5969. Para verificar su funcionamiento antes de realizar la transferencia del *bitstream*, se empleó la instrucción IDCODE, ya que esta permite seleccionar el registro de identificación del SoC FPGA y obtener su contenido a través de la cadena JTAG.

Debido a que el SoC FPGA está configurado en modo cascada, los controladores TAP y DAP están conectados en serie, el algoritmo debe considerar ambos registros de instrucciones durante cada operación. Puesto que la comunicación está dirigida exclusivamente al PL, donde el controlador TAP es quien gestiona esa zona, se carga la instrucción *IDCODE*, mientras que en el controlador DAP se introduce la instrucción *BYPASS*. De esta manera, el DAP permanece fuera de la operación de interés y se mantiene la continuidad de la cadena JTAG durante el acceso al registro de identificación de la PL.

En la figura 6.6 se observa la secuencia generada por el microcontrolador durante esta prueba. Mediante TCK y TMS se realiza el desplazamiento del controlador TAP entre los estados necesarios para acceder al registro de instrucciones. Una vez alcanzado el estado *shift-ir*, se desplaza la cadena de instrucciones correspondientes y posteriormente

se actualizan las instrucciones de ambos controladores. A continuación, el TAP se desplaza hacia *shift-dr*, donde el contenido del registro de identificación es enviado por TDO y recibido por el microcontrolador.

Como referencia para la prueba, el valor esperado del *IDCODE* correspondiente a la región PL fue obtenido previamente mediante el lenguaje de comandos de herramientas (TCL, por sus siglas en inglés) en *Vivado* y el valor es 0x23727093. El valor obtenido mediante la interfaz JTAG emulada coincidió con el reportado por *Vivado* para el dispositivo, lo que permite comprobar que el algoritmo realiza correctamente las transiciones del controlador TAP, el envío de instrucciones por TDI y la recepción de información por TDO. Por la tanto, esta prueba verifica la comunicación entre el MSP430FR5969 y el controlador TAP de la región PL del SoC FPGA antes de proceder con la transferencia del *bitstream*.

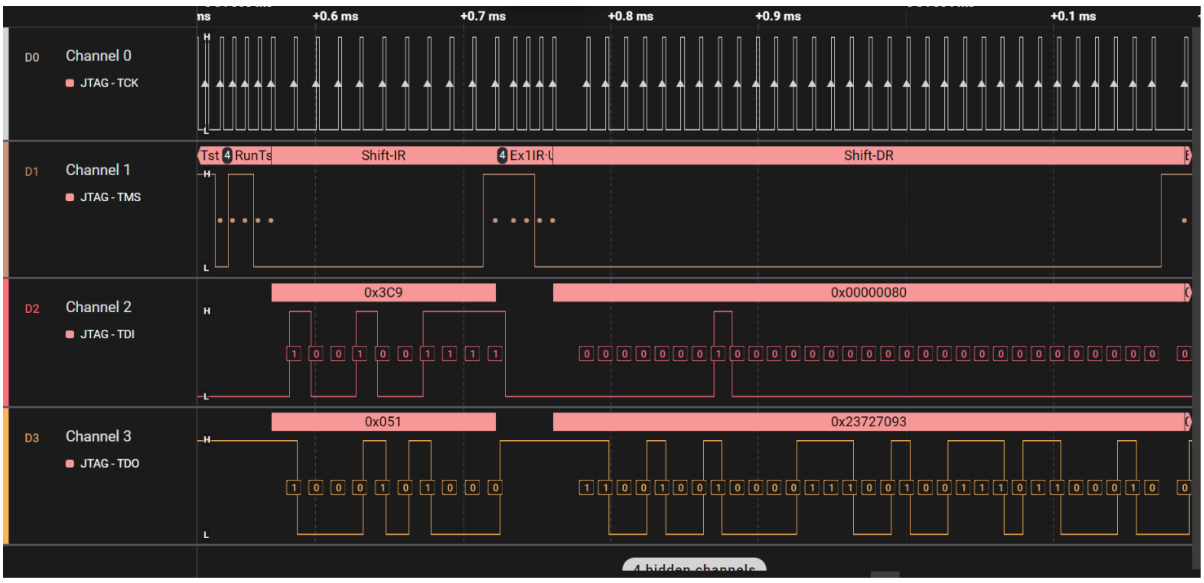


Figura 6.6: Prueba de comunicación exitosa entre el microcontrolador y el dispositivo SoC FPGA mediante la solicitud de *IDCODE*.

6.3. Transferencia del archivo *bitstream*

Una vez comprendida con mayor profundidad la estructura e información que se envía dentro del *bitstream*, y de haber entablado comunicación con el dispositivo SoC FPGA desde el microcontrolador por medio de la emulación de JTAG, el siguiente paso fue realizar la transferencia exitosa del archivo *bitstream*. Dicha transferencia se lleva a cabo desde una memoria hasta llegar al dispositivo de interés, todo eso con un microcontrolador como intermediario y encargado del proceso de transmisión.

El modelo del dispositivo SoC FPGA se configura desde *Vivado* por medio de su puerto TAP a través de un *USB-Blaster*. Sin embargo, como se aprecia en la figura 6.7, el *USB-Blaster* no se encuentra conectado. Por lo tanto, el dispositivo SoC FPGA fue configurado directamente por medio del microcontrolador, lo que permite verificar de forma funcional la transferencia del archivo *bitstream* a través del método propuesto.

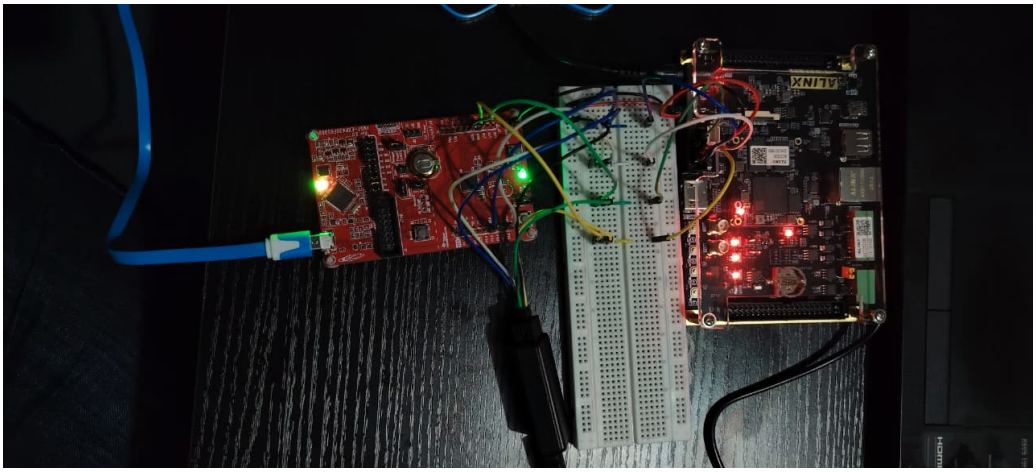


Figura 6.7: Transferencia exitosa del archivo *bitstream*.

El analizador lógico empleado, perteneciente a *Saleae*, captura en tiempo real la trama JTAG emulada por SPI + GPIO enviada por el microcontrolador durante el proceso de transferencia. Como se muestra en la figura 6.8, la información binaria adquirida mediante dicho decodificador confirma el envío de las señales y la secuencia de comunicación establecida. Bajo las condiciones empleadas durante la prueba, la transferencia completa del archivo *bitstream* tuvo una duración de aproximadamente 62 s. De esta manera se verificó el funcionamiento del sistema de transferencia implementado.

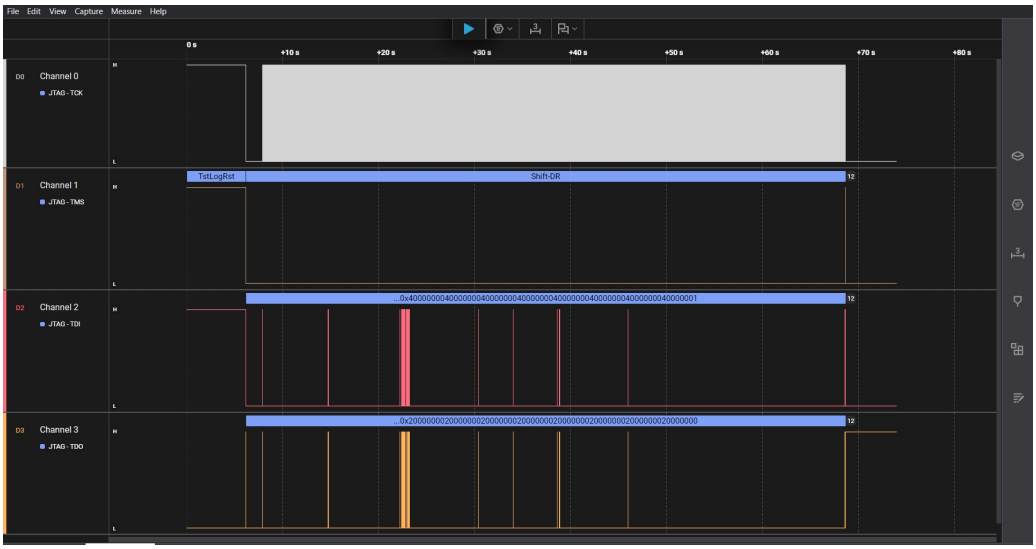


Figura 6.8: Transferencia del archivo *bitstream* capturada a través del decodificador.

Para el desplazamiento entre los diferentes estados del controlador TAP (figura 2.2) y para el envío de instrucciones hacia el SoC FPGA, se emplean señales generadas por GPIO con el fin de lograr una transición segura y precisa al siguiente estado deseado, así como un control adecuado del número de bits enviados a través del registro de instrucciones, como se muestra en la figura 6.9. Por otro lado, para el envío de datos a través del registro de datos, se cambia al modo SPI para acelerar la transferencia de bytes del *bitstream*, como se muestra en la figura 6.10.

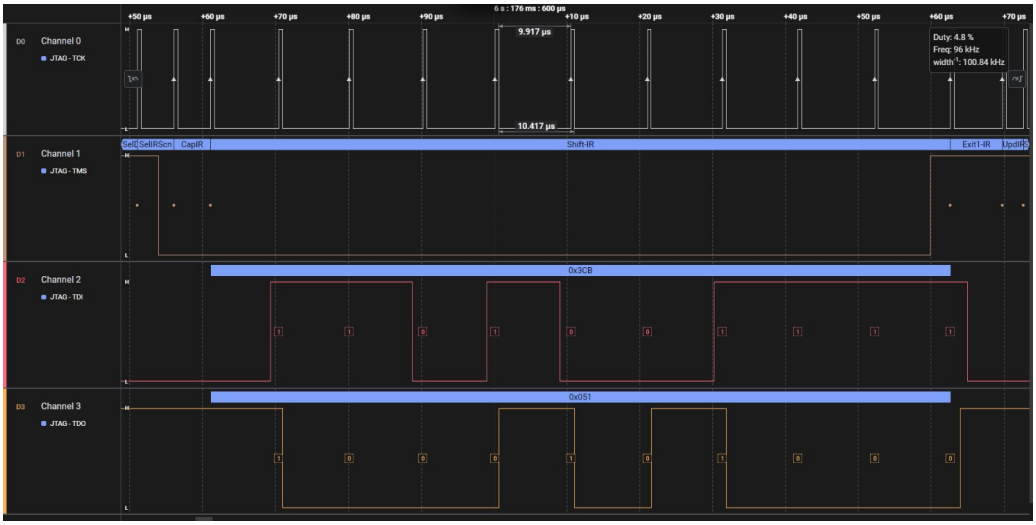


Figura 6.9: Desplazamiento entre los estados del controlador TAP y envío de instrucciones hacia el SoC FPGA mediante GPIO.

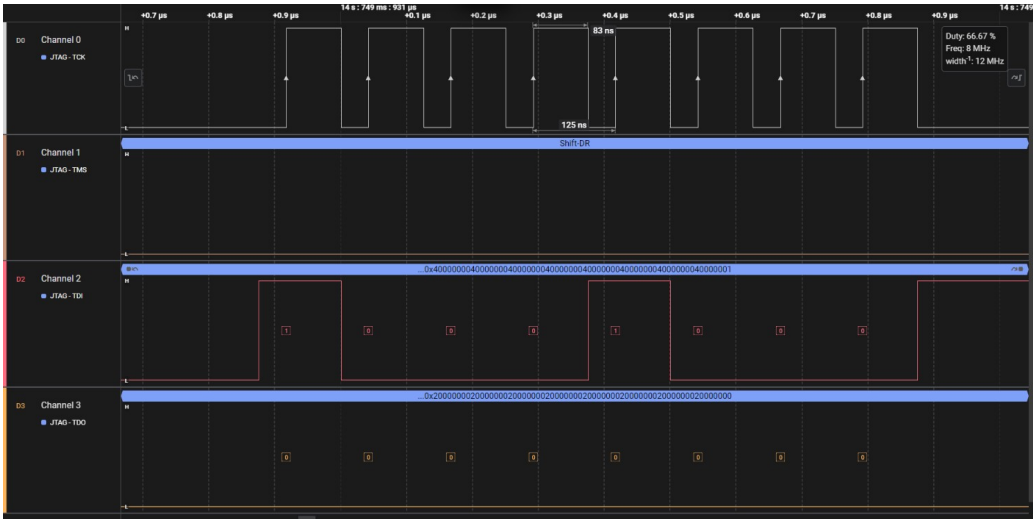


Figura 6.10: Transferencia del *bitstream* en bytes mediante SPI.

Capítulo 7

Conclusiones

En el presente trabajo se desarrolló el diseño e implementación de un sistema de reconfiguración externa para la región de la lógica programable de un SoC FGPA *Xilinx Zynq*-7000 (XC7Z020), empleando un microcontrolador MSP430FR5969 como elemento de control. El sistema fue diseñado para gestionar la información de configuración y transmitir el *bitstream* hacia el SoC FPGA mediante una interfaz JTAG emulada por software. Las pruebas realizadas permitieron validar funcionalmente esta propuesta bajo condiciones de laboratorio, de acuerdo con el alcance establecido para el trabajo.

Como resultado del proceso de diseño se definió una arquitectura compuesta por dos subsistemas principales: almacenamiento, con alto nivel de fiabilidad, y uno de control. Para el primero se seleccionó la memoria STT-MRAM EM064LXQADG13ES2T, debido principalmente a que satisface el requisito de incorporar tecnología resistente a la radiación. Para el subsistema de control se seleccionó el MSP430FR5969 a partir del análisis comparativo de alternativas, considerando sus recursos de comunicación, disponibilidad comercial, memoria FRAM y antecedentes documentados de utilización en sistemas espaciales.

Asimismo, se desarrolló una herramienta auxiliar para el análisis del *bitstream* en formato RBT. Mediante esta herramienta fue posible identificar e interpretar diferentes elementos de la estructura del archivo de configuración, entre ellos encabezados, comandos, registros, operaciones y datos. Su implementación permitió disponer de una representación interpretable de la información generada para dispositivos de la serie 7 de *Xilinx* y sirvió como apoyo para el desarrollo del algoritmo de reconfiguración.

Respecto a la interfaz de configuración, se implementó la emulación de JTAG desde el MSP430FR5969. La comunicación entre el microcontrolador y el SoC FPGA se verificó mediante la solicitud y recepción del *IDCODE*, con lo cual se comprobó que el algoritmo desarrollado era capaz de controlar las señales de la interfaz JTAG y obtener una respuesta válida del dispositivo. Esta prueba permitió validar de manera independiente la comunicación antes de realizar la transferencia completa del *bitstream*.

Posteriormente, se implementó la transferencia del *bitstream* desde la fuente de almacenamiento hacia el SoC FPGA, utilizando al microcontrolador como intermediario del proceso. Durante la implementación, las transiciones del controlador TAP y el envío de

instrucciones se realizaron mediante GPIO, mientras que el módulo SPI se empleó para la transferencia de los bytes del *bitstream*. Las señales obtenidas fueron observadas mediante un analizador de estados lógicos, lo que permitió verificar la secuencia de control y la transmisión de los datos de configuración. Asimismo, la configuración del SoC FPGA se realizó mediante el microcontrolador sin emplear el programador *USB-Blaster* utilizado normalmente durante el desarrollo.

En conjunto los resultados obtenidos permiten concluir que se cumplieron los objetivos específicos relacionados con el análisis del *bitstream*, la implementación de una interfaz JTAG por *software*, la transferencia del archivo de configuración y el diseño de detalle del sistema. De esta manera, se comprobó experimentalmente la viabilidad funcional, bajo condiciones de laboratorio, de utilizar un microcontrolador externo para controlar el proceso de configuración de la región PL de un SoC FPGA *Xilinx Zynq*-7000 mediante JTAG emulado.

7.1. Trabajo a futuro

Como trabajo a futuro, se propone extender el sistema de reconfiguración desarrollado para que no solo realice modificaciones en la región PL del dispositivo SoC FPGA, sino que también pueda intervenir en la zona PS. En el alcance actual de este trabajo se realizó el diseño y la implementación del sistema de reconfiguración y se verificó la reconfiguración de la región PL, por medio de un microcontrolador que emula la comunicación por JTAG a través del puerto TAP.

El objetivo principal de esta etapa futura será lograr la carga y ejecución de *software* en la zona PS mediante el mismo principio de operación empleado para la región PL; es decir, por medio del uso de un microcontrolador externo como entidad de control y el puerto TAP como medio de acceso por JTAG. Para alcanzar este objetivo, se plantea primero caracterizar el flujo oficial de programación de la zona PS mediante herramientas como *Vitis*, con el fin de identificar las operaciones necesarias para cargar *software* hacia el sistema de procesamiento.

Una vez comprendido el flujo oficial de programación, el siguiente objetivo será definir una secuencia mínima que se pueda implementar desde el microcontrolador. Para ello, se determinará el tipo de archivo que será manipulado; además, de analizar las condiciones de inicialización requeridas por el PS, como memoria, periféricos básicos y estado del procesador antes de ejecutar una aplicación cargada externamente.

Finalmente, el objetivo de integración será verificar que el microcontrolador pueda cargar y ejecutar aplicaciones en la zona PS, y posteriormente coordinar este proceso con la reconfiguración de la región PL. Con ello, el sistema reconfigurador podría evolucionar hacia una plataforma capaz de actualizar tanto la lógica programable como el *software* del sistema de procesamiento, lo que aumentará la flexibilidad operativa de la carga útil y abrirá la posibilidad de implementar esquemas de recuperación, actualización o tolerancia a fallas en aplicaciones espaciales bajo el estándar *CubeSat*.

Bibliografía

[1] NASA CubeSat Launch Initiative (CSLI), «CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers,» National Aeronautics y Space Administration, Moffett Field, CA, USA, inf. téc., 2017.

[2] C. Cappelletti, S. Battistini y B. K. Malphrus, eds., *CubeSat Handbook: From Mission Design to Operations*, 1.ª ed. London, U.K.: Academic Press (Elsevier), 2021, ISBN: 978-0-12-817884-3.

[3] C. A. L. Tenorio, «Diseño de un sistema de comando y manejo de información (SCMI) para nanosatélites bajo el estándar CubeSat 3U,» Tesis de licenciatura, Facultad de Ingeniería, Universidad Nacional Autónoma de México, Ciudad de México, México, 2025.

[4] Y. Sun, L. Guo, Y. Wang, Z. Ma e Y. Niu, «Fault diagnosis for space utilisation,» *The Journal of Engineering*, vol. 2019, n.º 23, págs. 8770-8775, 2019. DOI: 10.1049/joe.2018.9102

[5] R. D. Rasmussen, «Goal-Based Fault Tolerance for Space Systems Using the Mission Data System,» en *Proceedings of the IEEE Aerospace Conference*, IEEE, Big Sky, MT, USA, mar. de 2001, págs. 1-18.

[6] The Royal Society of Victoria. «Space Radiation.» [En línea]. Consultado: 26 de octubre de 2025. dirección: https://www.rsv.org.au/articles/space-radiation

[7] National Aeronautics and Space Administration (NASA). «Why Space Radiation Matters.» [En línea]. Consultado: 26 de octubre de 2025. dirección: https://www.nasa.gov/missions/analog-field-testing/why-space-radiation-matters/

[8] H. Austin et al., «Real-Time, Dynamically Reconfigurable Coprocessor for Image Processing in Small Satellites,» en *2025 IEEE Space Computing Conference (SCC)*, 2025, págs. 129-135. DOI: 10.1109/SCC66396.2025.00021

[9] T. Kuwahara, «FPGA-based Reconfigurable On-board Computing Systems for Space Applications,» Faculty of Aerospace Engineering and Geodesy, Tesis doct., Universität Stuttgart, Stuttgart, Germany, 2010.

[10] J. S. George, «An Overview of Radiation Effects in Electronics,» en *AIP Conference Proceedings*, vol. 2160, AIP Publishing, 2019, pág. 060 002. DOI: 10.1063/1.5127719

[11] Fortinet. «Fault Tolerance — Definition and Meaning.» [En línea]. Consultado: 26 de octubre de 2025, Fortinet CyberGlossary. dirección: https://www.fortinet.com/resources/cyberglossary/fault-tolerance

[12] I. Mcloughlin, C. Leong y C. Yeo, «Fault Tolerant Computer for Low Earth Orbit Micro Satellites,» en *Proceedings of the 4th IAA Symposium on Small Satellites for Earth Observation (ESA SP-571)*, European Space Agency (ESA), Berlin, Germany, mayo de 2003.

[13] R. W. Butler, «A Primer on Architectural Level Fault Tolerance,» NASA Langley Research Center, Hampton, Virginia, USA, Technical Memorandum NASA/TM-2008-215108, feb. de 2008.

[14] Departamento de Informática, Universidad de Valladolid, «Sistemas tolerantes a fallos,» Universidad de Valladolid, Valladolid, España, inf. téc., n.d. Capítulo 6 del documento académico sobre arquitectura de sistemas digitales.

[15] A. Cratere, L. Gagliardi, G. A. Sanca, F. Golmar y F. Dell'Olio, «On-board computer for CubeSats: State-of-the-art and future trends,» *IEEE Access*, vol. 12, págs. 99 537-99 571, 2024. DOI: 10.1109/ACCESS.2024.3428388

[16] National Aeronautics and Space Administration, «Mitigating Risks of Single-Event Effects in Space Applications,» NASA Engineering y Safety Center, inf. téc. Technical Bulletin No. 19-01-1, dic. de 2021, Dec. 3, 2021.

[17] AMD, *Zynq-7000 SoC Technical Reference Manual*, v1.12, UG585, Jan. 13, 2020, Advanced Micro Devices, Inc., 2020.

[18] CoinMarketCap, *Bitstream*, https://coinmarketcap.com/academy/glossary/bitstream, [Consultado: 09-dic-2025].

[19] AMD, *7 Series FPGAs Configuration User Guide*, UG470 (v1.17), Dec. 5, 2023, Advanced Micro Devices, Inc., 2023.

[20] HardwareBee, *FPGA Bitstream*, https://hardwarebee.com/WikiBee/fpga-bitstream/, [Consultado: 09-dic-2025].

[21] AMD, *Vivado Design Suite User Guide: Dynamic Function eXchange*, v2023.1, UG909, Advanced Micro Devices, Inc., 2023.

[22] XJTAG Ltd., *¿Qué es JTAG y cómo puedo usarlo?* XJTAG Ltd., Cambridge, Reino Unido, 2020.

[23] R. Singh, D. Singh y N. S. Chaudhari, «A Tool for Debugging JTAG,» *International Journal of Computer Applications*, vol. 975, págs. 8887-8890, 2015.

[24] *IEEE Standard for Test Access Port and Boundary-Scan Architecture*, IEEE Computer Society, 2013.

[25] K. Lee, Y. Lee, H. Lee y K. Yim, «A Brief Review on JTAG Security,» en *Proc. 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, IEEE, 2016, págs. 486-490. DOI: 10.1109/IMIS.2016.102

[26] Punto Flotante S.A., *Tutorial estándar protocolo de comunicación SPI: protocolo síncrono de comunicaciones MOSI, MISO, SCK*, https://puntoflotante.net/COMUNICACION-SPI-TUTORIAL.htm, [En línea]. Consultado: 25-sept-2025.

[27] P. Dhaker, *Introduction to SPI Interface*, https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html, [En línea]. Consultado: 25-sept-2025.

[28] Chengdu Ebyte Electronic Technology Co., Ltd., *Explicación detallada del protocolo de comunicación SPI*, https://www.es-ebyte.com/news/470, [En línea]. Consultado: 25-sept-2025, jul. de 2023.

[29] Texas Instruments, *MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's Guide*, SLAU367P, User's Guide, Originally published Oct. 2012, Texas Instruments, abr. de 2020.

[30] Microchip Technology Inc. «SPI Slave Interface,» Microchip Technology Inc., visitado 26 de ago. de 2026. dirección: https://onlinedocs.microchip.com/oxy/GUID-99D87AEC-CC09-454B-A204-1ACBCFB73834-en-US-2/GUID-667224EB-6DA6-4BE2-B522-AABAB8BA3971.html

[31] S. Burrell, *Comprensión de la interfaz SPI: guía para principiantes*, https://wraycastle.com/es/blogs/knowledge-base/what-is-an-spi-interface, [En línea]. Consultado: 28-sept-2025, nov. de 2024.

[32] Ariat-Tech.com, *¿Qué es SPI? Interfaz periférica en serie explicada*, https://www.ariat-tech.es/blog/what-is-spi-serial-peripheral-interface-explained.html, [En línea]. Consultado: 28-sept-2025.

[33] Crystal Rugged. «Volatile memory vs non-volatile memory.» Accessed: 1-Mar-2026. dirección: https://www.crystalrugged.com/knowledge/volatile-memory-vs-non-volatile-memory/

[34] VPN Unlimited. «Memoria volátil.» Accessed: 1-Mar-2026. dirección: https://www.vpnunlimited.com/es/help/cybersecurity/volatile-memory

[35] VPN Unlimited. «Memoria no volátil.» Accessed: 1-Mar-2026. dirección: https://www.vpnunlimited.com/es/help/cybersecurity/non-volatile-memory

[36] PediaExpertos. «Diferencia entre SRAM y DRAM.» Accessed: 3-Mar-2026. dirección: https://pediaexpertos.com/diferencia-entre-sram-y-dram/

[37] GuíaHardware. «Cómo funciona la memoria RAM.» Accessed: 3-Mar-2026. dirección: https://www.guiahardware.es/como-funciona-memoria-ram/

[38] GeeksforGeeks. «Difference between EPROM and EEPROM.» Accessed: 5-Mar-2026. dirección: https://www.geeksforgeeks.org/digital-logic/difference-between-eprom-and-eeprom/

[39] TechTarget. «EPROM (erasable programmable read-only memory).» Accessed: 5-Mar-2026. dirección: https://www.techtarget.com/whatis/definition/EPROM

[40] mari, *Memory*, SlideServe (online presentation), Accessed: 2026-03-05, ago. de 2014. dirección: https://www.slideserve.com/mari/memory

[41] Profesional Review. «NOR Flash: qué es y cómo funcionan estas memorias.» Accessed: 5-Mar-2026. dirección: https://www.profesionalreview.com/2026/03/01/nor-flash-que-es-y-como-funcionan-estas-memorias/

[42] GeeksforGeeks. «Differences between NAND vs NOR Flash Memory.» Accessed: 5-Mar-2026. dirección: https://www.geeksforgeeks.org/electronics-engineering/differences-between-nand-vs-nor-flash-memory/

[43] ReversePCB. «Memoria Flash NOR y memoria Flash NAND.» Accessed: 5-Mar-2026. dirección: https://reversepcb.com/es/memoria-flash-nor-y-memoria-flash-nand/

[44] IC Components. «What is ferroelectric RAM (FRAM): history, uses, and features.» Accessed: 6-Mar-2026. dirección: https://www.ic-components.es/blog/what-is-ferroelectric-ram(fram)-history,uses,features.jsp

[45] K. Shirriff. «Ramtron ferroelectric FRAM die analysis.» Accessed: 6-Mar-2026. dirección: https://www.righto.com/2024/09/ramtron-ferroelectric-fram-die.html

[46] Softtek. «Memoria MRAM: del almacenamiento al procesamiento de datos.» Accessed: 5-Mar-2026. dirección: https://blog.softtek.com/es/memoria-mram-del-almacenamiento-al-procesamiento-de-datos

[47] IMEC. «MRAM technologies for space applications: unified cache memory.» Accessed: 5-Mar-2026. dirección: https://www.imec-int.com/en/articles/mram-technologies-space-applications-unified-cache-memory

[48] Z. Pastuovic et al., «Single event upset cross section and total ionizing dose in simple commercial-off-the-shelf microelectronic devices using advanced proton, X-ray, and gamma radiation sources,» *European Physical Journal Plus*, vol. 140, pág. 557, 2025. DOI: 10.1140/epjp/s13360-025-06494-8

[49] A. Bacchini, G. Furano, M. Rovatti y M. Ottavi, «Total Ionizing Dose Effects on DRAM Data Retention Time,» *IEEE Transactions on Nuclear Science*, vol. 61, n.º 6, págs. 3690-3693, dic. de 2014. DOI: 10.1109/TNS.2014.2365532

[50] M. Vujisic, K. Stankovic y A. Vasic, «Comparison of gamma ray effects on EPROMs and E2PROMs,» *Nuclear Technology & Radiation Protection*, vol. 24, n.º 1, págs. 61-67, 2009. DOI: 10.2298/NTRP0901061V

[51] D. L. Hansen, R. Hillman, F. Meraz, J. Montoya y G. Williamson, «Radiation Performance of a Flash NOR Device,» en *2018 IEEE Nuclear and Space Radiation Effects Conference (NSREC)*, 2018, págs. 154-158. DOI: 10.1109/NSREC.2018.8584289

[52] M. J. Marinella, «Radiation Effects in Advanced and Emerging Nonvolatile Memories,» *IEEE Transactions on Nuclear Science*, vol. 68, n.º 5, págs. 546-572, mayo de 2021. DOI: 10.1109/TNS.2021.3074139

[53] R. Ladbury, T. Wilcox, A. Phan y K. A. LaBel, «Single-Event Effects in Peripheral Circuitry of a Commercial Ferroelectric RAM (FRAM),» en *IEEE Radiation Effects Data Workshop (REDW)*, 2011.

[54] HIREX Engineering, «Single Event Effects Radiation Test Report: Part Type CY7C1049B, 4 Mb SRAM, Manufacturer: Cypress,» HIREX Engineering, inf. téc. HRX/SEE/0072, Issue 01, dic. de 2002, European Space Agency Contract Report.

[55] M. Herrmann, K. Grürmann y F. Gliem, «Heavy Ion SEE Test of 4 Gbit DDR3 SDRAM Devices,» Institut für Datentechnik und Kommunikationsnetze, Technische Universität Braunschweig, inf. téc. TN-IDA-RAD-13/1A, nov. de 2014, Test report, ESTEC Contract No. 4000101358/10/NL/AF.

[56] S. S. Dahiwale, N. S. Shinde, D. Kanjilal, V. N. Bhoraskar y S. D. Dhole, «Swift heavy ion induced single event upsets in high density UV-EPROMs,» *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 266, págs. 1729-1733, 2008. DOI: 10.1016/j.nimb.2007.12.047

[57] D. Chen et al., «Single-Event Effect Performance of a Conductive-Bridge Memory EEPROM,» Presented at the IEEE Nuclear and Space Radiation Effects Conference (NSREC), Boston, MA, USA, Jul. 13–17, 2015, 2015.

[58] NTChip. «SRAM vs DRAM: What is the difference between SRAM and DRAM?» Accessed: 6-Mar-2026. dirección: https://www.ntchip.com/electronics-news/sram-vs-dram

[59] ICRFQ. «EPROM vs EEPROM: key differences, working principles and applications.» Accessed: 6-Mar-2026. dirección: https://www.icrfq.com/blog/EPROM-vs-EEPROM.html

[60] S. S. P. Parkin y S. Yang, «Memory on the racetrack,» *Nature Nanotechnology*, vol. 10, n.º 3, págs. 195-198, 2015, Accessed: 6-Mar-2026. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC4182445/

[61] S. Hamsa, A. G. Ananth y N. Thangadurai, «A Study of Semiconductor Memory Technology by Comparing Volatile and Non-Volatile Memories,» *Journal of Advanced Research in Dynamical and Control Systems*, vol. 10, n.º 04-Special Issue, págs. 1252-1258, 2018.

[62] CORHA, *Radiation Hardness Assurance (RHA) Final Presentation*, Technical presentation, Radiation Hardness Assurance study, 2019.

[63] Avalanche Technology. «MRAM's immunity to radiation makes it a natural for space.» Accessed: 22-Feb-2026. dirección: https://www.avalanche-technology.com/mrams-immunity-to-radiation-makes-it-a-natural-for-space/

[64] B. A. Dahl et al., «Radiation Evaluation of Ferroelectric Random Access Memory Embedded in 180nm CMOS Technology,» Texas Instruments Incorporated, Dallas, TX, USA, inf. téc., 2016.

[65] M. Elghefari y S. McClure, «Radiation Effects Assessment of MRAM Devices,» Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, inf. téc. JPL Publication 08-19, mayo de 2008.

[66] L. Yu et al., «Design and implementation of on-board high reliability FPGA on-orbit reconfiguration,» *Measurement and Control*, vol. 59, n.º 3, págs. 302-314, 2026. DOI: 10.1177/00202940251323416

[67] F. Viel y C. A. Zeferino, «A Module for Remote Reconfiguration of FPGAs in Satellites,» en *Workshop IBERCHIP*, 2017.

[68] S. L. Eine, D. D. Langer, R. Birkeland y M. Orlandić, «Reconfiguration of FPGA during operation of small satellite for flexible hyperspectral data compression,» en *75th International Astronautical Congress (IAC)*, Paper IAC-24-B.4.6B.8, Milan, Italy, oct. de 2024.

[69] B. Fiethe, H. Michalik, C. Dierker, B. Osterloh y G. Zhou, «Reconfigurable system-on-chip data processing units for space imaging instruments,» en *2007 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2007, págs. 977-982. DOI: 10.1109/DATE.2007.364420

[70] Microchip Technology Inc., *RTAX-S Radiation-Tolerant FPGAs*, [En línea]. Disponible en: https://www.microchip.com/en-us/products/fpgas-and-plds/radiation-tolerant-fpgas/rtax-s. [Consultado: 28-abr-2026].

[71] P. Rangababu, «FPGA based System on Chip (SoC) for Space computation,» *PLANEX*, vol. 6, n.º 1, págs. 16-20, ene. de 2016.

[72] M. Nguyen, *FPGA Products Update*, Presentation, Microsemi Space Forum Russia, nov. de 2013.

[73] National Aeronautics and Space Administration, *Reconfigurable FPGA Technology*, 500-PG-8700.2.7B, NASA, 2012. visitado 30 de ago. de 2026. dirección: https://soma.larc.nasa.gov/lws/pdf_files/3.17%20500-PG-8700%202%207%20B%20Admin%20Ext%202%20%282019-20%29%20%281%29.pdf

[74] K. T. Ulrich y S. D. Eppinger, *Diseño y desarrollo de productos*, 5.ª ed. México, D.F., México: McGraw-Hill/Interamericana Editores, S.A. de C.V., 2013.

[75] EDN Network, *Using and Selecting COTS Components for Space Applications*, Accessed: 2026-02-20, 2023. dirección: https://www.edn.com/using-and-selecting-cots-components-for-space-applications/

[76] P. Thanigai y W. Goh, «MSP430™ FRAM Quality and Reliability,» Texas Instruments, Application Report SLAA526A, mayo de 2014, Originally published March 2012, revised May 2014. dirección: https://www.ti.com/lit/an/slaa526a/slaa526a.pdf

[77] E. A. Montoya et al., «Immunity of Nanoscale Magnetic Tunnel Junctions with Perpendicular Magnetic Anisotropy to Ionizing Radiation,» *Scientific Reports*, vol. 10, pág. 10 220, 2020. DOI: 10.1038/s41598-020-67257-2

[78] Xilinx, Inc., *Zynq-7000 SoC: DC and AC Switching Characteristics*, DS187 (v1.12.2), Xilinx, Inc., San Jose, CA, USA, jul. de 2018.

[79] Microchip Technology Inc. «Getting Started With SuperFlash Technology,» Microchip Technology Inc., visitado 4 de sep. de 2026. dirección: https://www.microchip.com/en-us/products/memory/serial-and-parallel-flash/getting-started

[80] Infineon Technologies AG, *CYRS17B512, 512 Mb (64 MB) Serial NOR Flash SPI/QSPI, 3.3 V, Radiation Hardened*, 002-30115 Rev. *G, Infineon Technologies AG, mar. de 2025. visitado 4 de sep. de 2026. dirección: https://www.infineon.com/assets/row/public/documents/10/49/infineon-cyrs17b512-512-mb-64-mb-serial-nor-flash-spi-qspi-3-datasheet-en.pdf

[81] B. Macintosh, S. D'Amico, A. Koenig et al., «Miniature Distributed Occulter Telescope: mDOT,» Stanford University, mDOT Concept Study 80NSSC19K0142.

[82] European Cooperation for Space Standardization (ECSS), *Space product assurance — Radiation hardness assurance — EEE components*, ECSS-Q-ST-60-15C Rev.1, Standard, ECSS Secretariat, ESA-ESTEC, mar. de 2025.

[83] Infineon Technologies AG, *CYRS17B512: 512 Mb (64 MB) serial NOR flash, SPI/QSPI, 3.3 V, Radiation hardened*, 002-30115 Rev. *G, Datasheet, Infineon Technologies AG, mar. de 2025.

[84] Infineon Technologies AG, *CYRS17B512-133AZMB*, [En línea]. Disponible en: https://www.infineon.com/part/CYRS17B512-133AZMB. [Consultado: 27-mar-2026].

[85] Everspin Technologies, *MRAM for Radiation-hard Markets*, [En línea]. Disponible en: https://www.everspin.com/mram-radiation-hard-markets. [Consultado: 27-mar-2026].

[86] S. Vartanian y G. Allen, «PEAL Radiation Test Report: Single Event Effects (SEE) Evaluation,» Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, inf. téc., oct. de 2025.

[87] Everspin Technologies, *Expanded Serial Peripheral Interface (xSPI) Industrial STT-MRAM Datasheet: 8Mb, 16Mb, 32Mb, and 64Mb; SPI, QSPI and OSPI interfaces (STR & DTR)*, Datasheet, Version 3.2, Everspin Technologies, mayo de 2024.

[88] Mouser Electronics. «EM064LXQADG13ES2T Everspin Technologies,» Mouser Electronics, visitado 5 de sep. de 2026. dirección: https://www.mouser.com/ProductDetail/Everspin-Technologies/EM064LXQADG13ES2T

[89] Avalanche Technology, *High Performance Serial Persistent SRAM Memory: 1 Mbit–32 Mbit SPI (4-4-4) P-SRAM Memory*, Rev. D, jun. de 2019.

[90] Infineon Technologies AG, *16 Mb EXCELON Ultra Ferroelectric RAM (F-RAM), Serial (Quad SPI), 2048 K × 8, 108 MHz, Extended Industrial*, CY15B116QSN/CY15V116QSN, Datasheet 002-36371, Rev. *B, jun. de 2025.

[91] Mouser Electronics. «CY15B116QSN-108BKXQ Infineon Technologies,» Mouser Electronics, visitado 5 de sep. de 2026. dirección: https://www.mouser.com/ProductDetail/Infineon-Technologies/CY15B116QSN-108BKXQ

[92] RAMXEED Limited, *Memory FeRAM: 8 M (1024 K × 8) Bit Quad SPI MB85RQ8MX*, Datasheet DS501-00074-1v1-E, 2024.

[93] Mouser Electronics. «MB85RQ8MXPF-G-BCE1 RAMXEED,» Mouser Electronics, visitado 5 de sep. de 2026. dirección: https://www.mouser.com/en/ProductDetail/RAMXEED/MB85RQ8MXPF-G-BCE1

[94] RAMXEED Limited, *FeRAM Product List*, https://www.ramxeed.com/products/feram/feram-products.html, [Online]. Accessed: Sep. 5, 2026.

[95] H. Saito, *Research and Development of HZO FeRAM for Higher-Capacity FeRAM*, RAMXEED Limited [Online], Accessed: Sep. 5, 2026, mayo de 2026. dirección: https://www.ramxeed.com/tech-column/hzo-feram-development-for-high-capacity/

[96] Everspin Technologies, *Spin-transfer Torque MRAM Technology*, [Online], Accessed: Sep. 5, 2026. dirección: https://www.everspin.com/stt-mram-technology

[97] STMicroelectronics, *STM32StepByStep: STM32MCU basics*, [En línea]. Disponible en: https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:STM32MCU_basics. [Consultado: 26-mar-2026], nov. de 2025.

[98] Intel, *Basics of Programmable Logic*, [En línea]. Disponible en: https://www.intel.com/content/www/us/en/programmable/customertraining/webex/PLDBasics/presentation_html5.html. [Consultado: 26-mar-2026].

[99] J. Schneider e I. Smalley, *What is a field programmable gate array (FPGA)?* [En línea]. Disponible en: https://www.ibm.com/think/topics/field-programmable-gate-arrays. [Consultado: 26-mar-2026].

[100] M. Berg, «Field Programmable Gate Array (FPGA) Single Event Effect (SEE) Radiation Testing,» MEI Technologies, in support of NASA Goddard Space Flight Center, inf. téc., feb. de 2012, Prepared for NASA Electronic Parts and Packaging (NEPP) and Defense Threat Reduction Agency, IACRO #11-4395I.

[101] Texas Instruments, *MSP430FR5969*, *TI.com*, [En línea]. Disponible en: https://www.ti.com/product/MSP430FR5969?keyMatch=MSP430FR5969&tisearch=universal_search. [Consultado: 25-mar-2026].

[102] Texas Instruments, *MSP430FR5969-SP*, [En línea]. Disponible en: https://www.ti.com/product/MSP430FR5969-SP. [Consultado: 25-mar-2026].

[103] Texas Instruments, *MSP430FR5969-SP Radiation Hardened Mixed-Signal Microcontroller*, SLASEK0A, Datasheet, Revision A, Texas Instruments, mar. de 2018.

[104] VORAGO Technologies, *Extreme Temperature ARM Cortex-M0 Microcontroller VA10800*, Datasheet, Revision 2.3, VORAGO Technologies, 2019.

[105] VORAGO Technologies, *Easy Radiation-Hardening of Conventional Integrated Circuits Using HARDSIL® Technology*, White paper, VORAGO Technologies.

[106] J.-M. Armani, S. Blairon y A. Ureña-Acuña, «Total ionizing dose response of commercial off-the-shelf microcontrollers and operational amplifiers,» en *2020 20th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, Toulouse, France, oct. de 2020. DOI: 10.1109/RADECS50773.2020.9857725

[107] S. Gerardin et al., *CORHA Project: Radiation Screening of COTS Components and Verification of COTS RHA Approach – Part 2*, Presented at the TEC-QEC Final Presentation Days, ESA/ESTEC, Noordwijk, The Netherlands, Accessed: Sep. 6, 2026, jun. de 2023. dirección: https://indico.esa.int/event/444/contributions/8087/attachments/5303/8506/CORHA%20Part%202%20-%20Radiation%20screening%20of%20COTS%20components%20and%20verification%20of%20COTS%20RHA%20approach.pdf

[108] D. J. Barnhart et al., «Enabling Space Sensor Networks with PCBSat,» en *Proceedings of the 21st Annual AIAA/USU Conference on Small Satellites*, Paper SSC07-IV-4, Logan, UT, USA, 2007.

[109] T. Fried et al., «Radiation Testing of Low Cost, Commercial Off the Shelf Microcontroller Board,» *Nuclear Engineering and Technology*, vol. 53, n.º 10, págs. 3335-3343, 2021. DOI: 10.1016/j.net.2021.05.005

[110] J. Aarestad, P. Eaton, W. Burke, D. Alexander, J. Salzman y K. Avery, «Single Event Effects Testing of the Hardened Texas Instruments MSP430FR5739 Microcontroller on with Embedded Ferroelectric Memory,» en *2016 IEEE Radiation Effects Data Workshop (REDW)*, 2016. DOI: 10.1109/NSREC.2016.7891740

[111] Austrian Space Industry Association (AUSTROSPACE), «Austrospace Annual Report 2022,» Austrian Space Industry Association (AUSTROSPACE), Holzhausen, Austria, inf. téc., oct. de 2023.

[112] A. I. Chumakov, «An Evaluation of the Sensitivity of Integrated Circuits to Single Event Effects for the Point Charge Collection Area,» *Russian Microelectronics*, vol. 44, n.º 1, págs. 27-32, 2015. DOI: 10.1134/S1063739715010047

[113] Texas Instruments, *MSP430FR596x, MSP430FR594x Mixed-Signal Microcontrollers*, SLAS704G, Datasheet, Originally published Oct. 2012, Texas Instruments, ago. de 2018.

[114] STMicroelectronics, *STM32F103RG*, [Online], Accessed: Sep. 6, 2026. dirección: https://www.st.com/en/microcontrollers-microprocessors/stm32f103rg.html

[115] Mouser Electronics, *STM32F103RGT6 STMicroelectronics*, [Online], Accessed: Sep. 6, 2026. dirección: https://www.mouser.mx/es/ProductDetail/STMicroelectronics/STM32F103RGT6

[116] Atmel Corporation, *ATmega128/L Datasheet Summary*, Rev. 2467XS–AVR–06/11, jun. de 2011. dirección: https://ww1.microchip.com/downloads/en/DeviceDoc/2467S.pdf

[117] Mouser Electronics, *ATMEGA128-16AN Microchip Technology*, [Online], Accessed: Sep. 6, 2026. dirección: https://www.mouser.mx/es/ProductDetail/Microchip-Technology/ATMEGA128-16AN

[118] NXP Semiconductors, *Kinetis K66 Sub-Family: 180 MHz ARM Cortex-M4F Microcontroller*, Data Sheet K66P144M180SF5V2, Rev. 4, abr. de 2017. dirección: https://www.nxp.com/docs/en/data-sheet/K66P144M180SF5V2.pdf

[119] Mouser Electronics, *ARM Cortex-M4F Core, 32 bit, 180 MHz ARM Microcontrollers – MCU*, [Online], Accessed: Sep. 6, 2026. dirección: https://www.mouser.mx/es/c/semiconductors/embedded-processors-controllers/microcontrollers-mcu/arm-microcontrollers-mcu/

[120] National Aeronautics and Space Administration, «State-of-the-Art Small Spacecraft Technology,» Small Spacecraft Systems Virtual Institute, NASA Ames Research Center, Moffett Field, CA, USA, inf. téc. NASA/TP-2024-0001462, feb. de 2024.

[121] G. Li et al., «Lessons Learned from the Development and Flight of the First Miniature Tethered Electrodynamics Experiment (MiTEE-1),» en *Proceedings of the 35th Annual Small Satellite Conference*, Paper SSC21-WKIII-07, 2021.

[122] Polímetro, *FPGA: Qué es, cómo funciona, para qué sirve y todo lo que debes saber sobre estos chips reconfigurables*, [En línea]. Disponible en: https://www.polimetro.com/que-es-una-fpga/. [Consultado: 3-abr-2026], sep. de 2025.

[123] Xilinx, Inc., *Zynq-7000 SoC Data Sheet: Overview*, DS190, Product Specification, Version 1.11.1, Xilinx, Inc., jul. de 2018.

[124] *What is communication protocol?* Dirección: https://www.pubnub.com/learn/glossary/communication-protocols/

[125] Texas Instruments, «JTAG Technical Overview,» Texas Instruments Incorporated, inf. téc., 2001.

[126] IEEE Standards Association, *IEEE SA Quick Reference Guide: Standards Development Process*, Piscataway, NJ, USA, 2021.

[127] FUJITSU Semiconductor Limited, «FUJITSU Semiconductor FRAM,» FUJITSU Semiconductor Limited, Yokohama, Japan, inf. téc. AD05-00033-7E, mayo de 2014.

[128] Xilinx, Inc., *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide*, UG480 (v1.10.1), Xilinx, Inc., San Jose, CA, USA, oct. de 2016.

[129] K. A. LaBel, *Proton Single Event Effects (SEE) Guideline*, NASA Electronic Parts y Packaging (NEPP) Program, ago. de 2009.

[130] J. Heidecker, «MRAM Technology Status,» Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, inf. téc. JPL Publication 13-3, feb. de 2013, NASA Electronic Parts and Packaging (NEPP) Program.

[131] D. Turner, *CLB vs. CPLD vs. FPGA: Choosing the Right Logic for Your Embedded Design*, Microchip Technology Inc. [Online], Accessed: Sep. 6, 2026, jul. de 2026. dirección: https://www.microchip.com/en-us/about/media-center/blog/2026/clb-vs-cpld-vs-fpga