



Universidad de Valladolid



**ESCUELA DE INGENIERÍAS
INDUSTRIALES**

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

**Implementación basada en una FPGA de
un algoritmo para el test de pistas de
tarjetas de circuito impreso**

Autor:

Hidalgo Uña, Rodrigo

Tutor:

**Cáceres Gómez, Santiago
Departamento Tecnología Electrónica**

Valladolid, febrero 2017.

Resumen

Debido al incremento en la cantidad de componentes y a la reducción del espacio entre los mismos, el test de las tarjetas de circuito impreso cada vez es más complejo. Por ello se hace necesario encontrar sistemas alternativos a las tradicionales camas de clavos.

El estándar IEEE 1149.1[1] sienta las bases para sustituir las mismas por una arquitectura *boundary-scan*, incluida en los propios chips del circuito.

Este trabajo fin de grado está enfocado en identificar los problemas relacionados con la interconexión de pistas presentes en las tarjetas de circuito impreso. Concretamente en la detección de los denominados *stuck-at-zero* y *stuck-at-one*.

Previamente a realizar la implementación, se revisan todas las consideraciones que se han de tener presentes, así como la búsqueda de un algoritmo que permita identificar los defectos correctamente. También se diseña una placa de test que permite simular defectos eléctricos.

El algoritmo se implementa mediante VHDL en una FPGA.

Palabras clave

Boundary scan, test de pistas, diagnóstico de fallos, FPGA, JTAG.

Abstract

Due to the latest technology improvements, testing of printed circuit boards is becoming more complex, especially because of the surface mounting devices. Pins proximity makes it difficult for ATEs to test the PCB through bed of nails.

IEEE 1149.1 standard[1] sets the basis for replacing them with boundary scan architecture, included in the chips itself.

This document focuses into identifying which defaults related to nets interconnection take place in a certain board. More precisely, it will take into account stuck-at-zero and stuck-at-one faults.

Before performing the implementation, all the considerations which have to be present are reviewed. The research for an algorithm that allows a real identification of the defects is also exposed. A board which allows for electric faults simulation is also designed.

The implementation in a FPGA of an algorithm which takes all of this into account is done in VHDL.

Key words

Boundary scan architecture, interconnect test, fault diagnosis, FPGA, JTAG.

Índices

Índice de contenido

RESUMEN	1
ABSTRACT	3
ÍNDICES	5
ÍNDICE DE CONTENIDO	5
ÍNDICE DE TABLAS	7
ÍNDICE DE IMÁGENES	8
GLOSARIO DE TÉRMINOS Y ABREVIATURAS	10
1 INTRODUCCIÓN Y OBJETIVOS	13
1.1 MARCO DEL TRABAJO FIN DE GRADO	13
1.2 OBJETIVOS	15
1.3 ESTRUCTURA DE LA MEMORIA	16
2 MARCO CONCEPTUAL	17
2.1 CONOCIMIENTOS PREVIOS	17
2.1.1 REDES	17
2.1.2 DEFECTOS ELÉCTRICOS	21
2.1.3 ESTÁNDAR IEEE 1149.1	24
2.1.4 ALGORITMOS	33
2.2 PLATAFORMA DE DESARROLLO	43
2.3 PROGRAMAS EMPLEADOS	47
2.3.1 VIVADO	47
2.3.2 PROTEUS 8	49
3 PLACA DE PRUEBAS	51

3.1	CONCEPTO	51
3.2	PROTECCIÓN ANTE CORTOCIRCUITOS	53
3.3	CIRCUITOS CONSIDERADOS	58
3.4	LISTADO DE COMPONENTES	59
3.5	DISEÑO	60
3.6	EJECUCIÓN Y CONFIGURACIÓN	65
4	IMPLEMENTACIÓN	69
4.1	EL TEST A REALIZAR	69
4.1.1	VERIFICAR LA INTEGRIDAD DE LA ARQUITECTURA <i>BOUNDARY SCAN</i>	69
4.1.2	VERIFICAR LA INTERCONEXIÓN ENTRE PISTAS	73
4.2	DISEÑO DEL ALGORITMO A IMPLEMENTAR	75
4.3	DESARROLLO DEL ALGORITMO	77
4.4	RESULTADOS EXPERIMENTALES	91
5	ESTUDIO ECONÓMICO	93
5.1	RECURSOS EMPLEADOS	93
5.2	COSTES DIRECTOS	94
5.2.1	COSTES DE PERSONAL	94
5.2.2	COSTES DE AMORTIZACIÓN DE EQUIPOS Y PROGRAMAS	95
5.2.3	COSTES DERIVADOS DE OTROS MATERIALES	96
5.2.4	COSTES DIRECTOS TOTALES	96
5.3	COSTES INDIRECTOS	97
5.4	COSTES TOTALES	98
6	CONCLUSIONES	99
6.1	TRABAJO FUTURO	100
7	REFERENCIAS	101
8	ANEJOS	105

Índice de tablas

1 - Ejemplo de vectores de test	34
2 - Muestra de vectores de respuesta recibidos	35
3 - Ejemplo de vectores de test previos a un aliasing syndrome	36
4 - Muestra de vectores de respuesta posteriores a un aliasing syndrome	36
5 - Ejemplo de vectores de test previos a un confounding syndrome	37
6 - Ejemplo de vectores de respuesta posteriores a un confounding syndrome	37
7 - Muestra del Counting Sequence Algorithm	38
8 - Muestra del Modified Counting Sequence Algorithm	39
9 - Muestra del True/Complement Test and Diagnosis Algorithm	40
10 - Muestra de Walking One's Algorithm	41
11 - Comparación de los distintos algoritmos propuestos	42
12 - Configuraciones de pistas presentes en la placa	62
13 - Diagnóstico propuesto por[3]	75
14 - Registros estáticos empleados	77
15 - Registros desplazadores empleados	77
16 - Contadores empleados	77
17 - Días efectivos estimados	94
18 - Total de horas empleadas en el trabajo fin de grado	95
19 - Amortización anual de los equipos y programas	95
20 - Otros costes directos	96
21 - Costes indirectos	97

Índice de imágenes

1 - Buffer	17
2 - Redes sencillas o simples	18
3 - Red con múltiples buffers de salida	18
4 - Red con múltiples buffers de entrada	19
5 - Red con múltiples buffers, o cluster	19
6 - Buffer triestado (B)	20
7 - Ejemplo de defecto Stuck-at-one	21
8 - Ejemplo de red Stuck-at-zero	22
9 - Ejemplo de red con defecto open fault	22
10 - Comportamiento de un defecto OR-type short	22
11 - Comportamiento de un defecto AND-type short	23
12 - Comportamiento de un defecto Strong-driven short	23
13 - Máquina de estados del puerto de acceso para test	26
14 - Muestra de una celda boundary scan. Celda BC_2 de [1]	29
15 - Descripción de puertos lógicos en un fichero BSDL	31
16 - Distribución física de los pines en un fichero BSDL	31
17 - Definición de la interfaz del TAP en un fichero BSDL	31
18 - Longitud del registro de instrucciones en un fichero BSDL	32
19 - Códigos binarios de instrucciones en un fichero BSDL	32
20 - Código cargado en CAPTURE IR en un fichero BSDL	32
21 - Descripción del registro boundary-scan en un fichero BSDL	32
22 - Basys 3 [12]	43
23 - Puerto PMOD en [12]	44
24 - Display 4 dígitos de 7 segmentos en [12]	45
25 - Interfaz gráfica de Vivado WebPack 2016.3	47
26 - Navegador de flujo de diseño	48
27 - Muestra de un diseño a nivel RTL	48
28 - Interfaz gráfica en Proteus 8	49
29 - Modo ISIS, ARES y 3D Visualizer en Proteus 8	50
30 - Posibles tipos de redes	51
31 - Defectos eléctricos deterministas	51
32 - Daños visibles provocados por cortocircuitos en circuitos integrados. Fuente [17]	53
33 - Daños internos de los circuitos integrados provocados por cortocircuitos. Fuente [17]	53
34 - Circuito MOSFET de protección propuesto por Banakar y Mathew en [17]	55
35 - Resumen de opciones para proteger un dispositivo del estrés térmico	55

36 - Simulación del fallo stuck-at one	56
37 - Simulación del fallo stuck-at zero	56
38 - Generación del fallo stuck open	56
39 - Simulación del fallo strong-driven short entre dos redes	57
40 - Simulación del fallo OR-type short entre dos redes	57
41 - Simulación del fallo AND-type short entre dos redes	57
42 - Lista de materiales	59
43 - Modelos de PIC32MX en Proteus 8	60
44 - Modelo de bloque empleado para provocar el defecto stuck-at	60
45 - Diseño de la placa en ARES	61
46 - Placa con los componentes emplazados. Vista del layout	63
47 - Visualización 3D de la placa	64
48 - Placa de test finalizada	65
49 - Unión entre la placa y la Basys 3	65
50 - Ejemplo de generación de fallos tipo short	66
51 - Muestra de generación de fallos tipo stuck-at	67
52 - Estados por los que pasa el TAP durante la verificación del puerto	70
53 - Representación del estado de los registros de instrucciones tras Capture IR	70
54 - Ejemplo de cadena boundary-scan	72
55 - Estados por los que pasa el TAP durante la verificación de la cadena boundary-scan	72
56 - Estados por los que pasa el TAP durante el test de defectos eléctricos	73
57 - Inicio del algoritmo	78
58 - Verificación del estado JTAG	79
59 - Verificación del estado boundary-scan	80
61 - Generación test stuck-at - parte 1	81
62 - Generación test stuck-at - parte 2	82
63 - Generación test stuck-at - parte 3	83
64 - Generación test stuck-at - parte 4	84
65 - Generación test stuck-at - parte 5	85
66 - Envío y análisis del vector de test - parte 1	86
67 - Envío y análisis del vector de test - parte 2	87
68 - Mostrar resultados del test	88
69 - Mostrar fallos tipo stuck-at-one	89
70 - Mostrar fallos tipo stuck-at-zero	90

Glosario de términos y abreviaturas

ATE: Automatic Test Equipment. Equipo para test automáticos. Dispositivo cuya función es la realización de ensayos sobre un dispositivo, equipo o unidad. De forma autónoma realiza una serie de mediciones y verifica los resultados.

Bed of nails: Véase cama de clavos.

BIST: Built-in Self-test. Test incluido en el propio dispositivo. Es un mecanismo que permite al propio dispositivo en que se implemente verificar su correcto funcionamiento.

Boundary scan: Arquitectura implementada en circuitos integrados para verificar los componentes y conexiones en las tarjetas de circuito impreso.

Cama de clavos: Accesorio de los equipos de test automáticos que verifica las conexiones y funcionamiento de ciertos circuitos mediante sondas.

Chip: Véase circuito integrado.

CI: Circuito integrado. Véase circuito integrado.

Circuito integrado: Componente de pequeñas dimensiones que contiene circuitos electrónicos protegidos por un encapsulado cerámico o plástico del que salen los conectores que permiten su conexión a la tarjeta de circuito impreso.

FPGA: Field Programmable Gate Array. Matriz de puertas lógicas programables. Dispositivo programable que contiene lógica cuya conexión y función es configurada mediante lenguajes de descripción *hardware*.

IC: Integrated Circuit. Véase circuito integrado.

PCB: Printed Circuit Board. Véase tarjeta de circuito impreso.

Tarjeta de circuito impreso: También denominada placa de circuito impreso, o simplemente placa. Es el soporte sobre el que se colocan y conectan los diferentes circuitos integrados y componentes que dan lugar a un diseño. Se emplea como referencia al conjunto una vez incorporados los componentes a la misma.

1 Introducción y objetivos

1.1 Marco del trabajo fin de grado

Son gran cantidad los dispositivos electrónicos que nos rodean en el día a día, todos ellos cuentan en su interior con algún tipo de circuitería encargada de realizar cierta función. Estos circuitos conforman la denominada tarjeta de circuito impreso.

La industria electrónica, antes de dar como apta una tarjeta, precisa realizar una serie de pruebas sobre la misma. El objetivo de estas pruebas es variado, pues hay que verificar múltiples características. Se han de tener en cuenta las dimensiones de la tarjeta, la posición de los componentes empleados, la conexión entre los mismos, el funcionamiento del conjunto acorde a las especificaciones requeridas...

Una muestra de los test que se realizan prácticamente en la totalidad de tarjetas sería la inspección automática mediante luz visible, la inspección automática mediante rayos X, el test funcional o el test estructural.

Toda placa ha de pasar por una serie de pruebas para verificar su correcto funcionamiento, pues la comercialización de unidades defectuosas puede llegar a tener un alto coste para la empresa, así como reducir su reputación. Es por ello que se busca poder realizar una verificación completa de las tarjetas a comercializar.

No obstante, el elevado precio de los equipos de test, así como el largo tiempo que requiere la realización de un test completo a cada una de las tarjetas, hace imposible verificar en su totalidad todas las tarjetas. Por este motivo se realiza un seguimiento continuo de las placas durante el proceso de fabricación, seguido de un test funcional al final del proceso productivo. Este test verifica el correcto funcionamiento de la tarjeta como un conjunto, siendo incapaz de detectar fallos internos en la misma.

El test estructural se compone de múltiples pruebas relacionadas con los componentes internos de la placa. Entre ellas se encuentra el test eléctrico al

que se someten las pistas para comprobar que las conexiones entre los componentes son las adecuadas.

Tradicionalmente este test lo realizan equipos de test automático (ATE) por medio de camas de clavos. Este método presenta varios inconvenientes, entre ellos: para cada diseño a probar se necesita modificar tanto su estructura física como la programación; el diseño de la placa ha de proporcionar acceso físico a los puntos de test.

El estándar IEEE 1149.1[1] presenta una serie de circuitería que añadida a un circuito integrado facilita el test, mantenimiento y soporte de dichos dispositivos. Este estándar está siendo ampliamente adoptado en la industria, especialmente en aquellos componentes que cuentan con un gran número de conexiones como son los circuitos programables o microprocesadores.

La gran acogida que ha tenido el estándar permite prescindir de las camas de clavos y parte de los equipos de test automático para la realización del test eléctrico, pues sus funciones se incluyen en los propios integrados.

Este trabajo final de grado aborda la realización del test eléctrico de pistas desde la propia tarjeta de circuito impreso a comprobar, prescindiendo de los equipos de test externos. Esto permite que la placa realice dicho test no solo previamente a su comercialización, sino también una vez se encuentre en funcionamiento, permitiendo así la detección de posibles fallos durante su vida útil.

1.2 Objetivos

Previamente a la realización del test eléctrico es necesario realizar una investigación sobre las materias expuestas a continuación.

- Valorar todas las configuraciones de redes presentes en las tarjetas de circuito impreso.
- Identificar los posibles defectos eléctricos que puedan afectar tanto a una pista aislada como a un conjunto de las mismas.
- Buscar un algoritmo generador de vectores de test eléctrico.

Una vez analizados estos temas, se hace preciso:

- Elegir un algoritmo generador de vectores de test.
- Diseñar y crear un soporte con el que generar los diferentes defectos eléctricos.
- Realizar la implementación del test eléctrico sobre la FPGA escogida. De tal manera que se identifique el defecto y su localización.
- Verificar la correcta implementación sobre el soporte creado.

Como podemos ver, tras la realización de todos estos objetivos nos encontraremos con una FPGA capaz de realizar el test eléctrico a una placa que permite generar diferentes defectos.

1.3 Estructura de la memoria

La presente memoria sigue un orden enfocado a la consecución de los objetivos presentados en el apartado previo.

En la primera parte del *capítulo 2 Marco conceptual*, se expone la investigación realizada sobre los considerados conocimientos previos, tales como la topología de redes, de defectos eléctricos, el estándar de test considerado, y los diferentes algoritmos existentes. A continuación, se detalla la plataforma empleada para realizar el trabajo fin de grado, así como los diferentes programas empleados.

El *capítulo 3 Placa de pruebas* desarrolla la creación del soporte generador de defectos eléctricos, desde su concepción hasta su creación.

En el *capítulo 4 Implementación* se detalla el algoritmo a implementar, así como los test previos que se han de realizar para asegurar que los resultados sean correctos.

A continuación, en el *capítulo 5 Resultados*, se analiza el resultado de la implementación sobre la placa de pruebas diseñada.

El *capítulo 6 Estudio económico* trata los diferentes costes asociados al desarrollo de este trabajo fin de grado, incluyendo personal, materiales y diferentes recursos.

Finalmente, para concluir la memoria, se presentan las conclusiones halladas, así como las referencias consultadas y los anexos.

2 Marco conceptual

2.1 Conocimientos previos

En este apartado se presenta una síntesis de la investigación realizada sobre los diferentes tipos de redes, los posibles defectos eléctricos, el estándar IEEE 1149.1 y los algoritmos existentes para la generación de vectores de test. Todos ellos elementos que se han de tener presentes previamente a la realización de un test eléctrico.

2.1.1 Redes

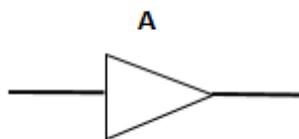
Un test eléctrico comprueba el estado en el que se encuentran las diferentes redes del circuito, es por ello que se hace necesario conocer qué es exactamente una red, su tipología, y algunos conceptos sobre las mismas.

2.1.1.1 Definición de red

En una placa de circuito impreso, una red se define como una superficie equipotencial, formada por un cable físico que conecta buffers de entrada con buffers de salida.[2]

En el caso de tarjetas de circuito impreso, las redes también se pueden denominar pistas.

Los buffers son dispositivos electrónicos encargados de separar su entrada de la salida, aportando una ganancia de 1, es decir no modifican la tensión.



1 - Buffer

En el presente documento se considera un buffer de entrada aquel que proporciona tensión a la red, y buffer de salida el que toma como referencia la tensión a la que está sometida la red.

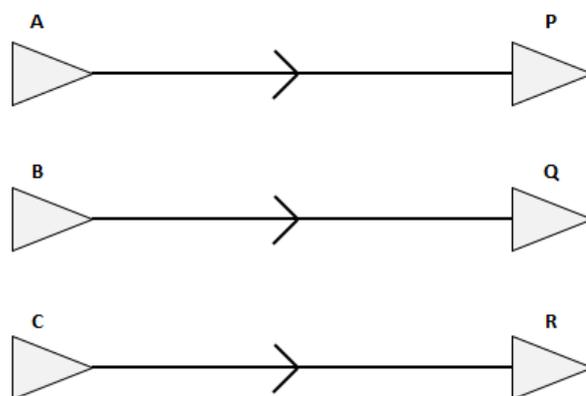
Los valores lógicos que puede tomar un buffer son 0 y 1.

2.1.1.2 Tipos de redes

Existen tantos tipos de redes como configuraciones en cuanto a buffers de entrada y salida, no obstante, todos ellos se pueden agrupar en los cuatro siguientes[3]:

- Redes sencillas o simples:

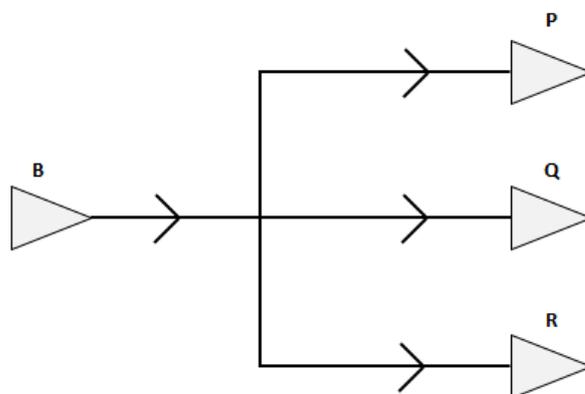
Este tipo de redes solo cuentan con un buffer de entrada y otro de salida, en la imagen siguiente se pueden ver 3 redes simples:



2 - Redes sencillas o simples

- Redes con múltiples buffers de salida:

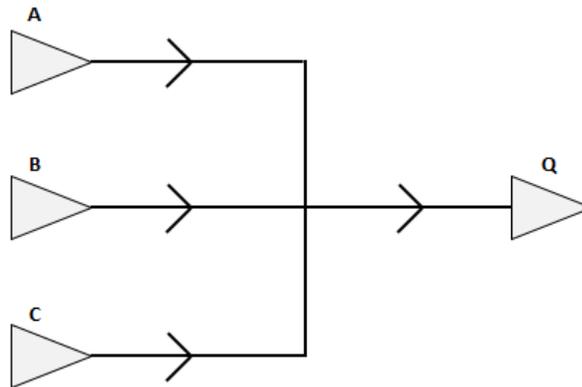
Estas redes cuentan con una única entrada conectada a varios buffers, todos ellos de salida. Todos los buffers de salida reciben el mismo valor de tensión. Este tipo de redes también se conocen como *fan-outs*.



3 - Red con múltiples buffers de salida

- Redes con múltiples buffers de entrada:

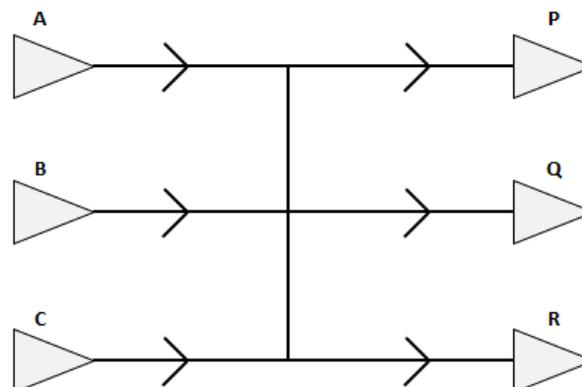
Esta clase de redes cuentan con varios buffers a la entrada, todos ellos conectados a una única salida. En estas redes, solo un buffer puede estar habilitado como salida, el resto han de encontrarse en alta impedancia. Este tipo de redes se conocen como *multiple drivers*.



4 - Red con múltiples buffers de entrada

- Redes con múltiples buffers, o cluster:

La configuración de red más compleja, incluye varios buffers tanto a la entrada como a la salida. Al igual que en el caso previo, solo puede estar activo uno de los buffers de entrada. Todos los buffers a la salida reciben el mismo valor de tensión. Esta configuración de red se conoce como *cluster*.

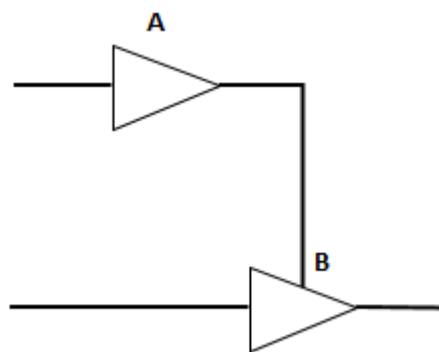


5 - Red con múltiples buffers, o cluster

2.1.1.3 Conceptos sobre redes

Cualquier red puede contar con un buffer triestado a su entrada, este tipo de buffers permiten un estado de alta impedancia a mayores de los estados lógicos 0 y 1. Un buffer entra en estado de alta impedancia cuando se quiere que su salida no afecte a la red en que se encuentra conectado. Habitualmente se encuentran presentes en buses y en puertos que permiten su funcionamiento tanto como entrada como salida.

En la siguiente imagen se puede ver un buffer triestado (B) gobernado por la salida del buffer (A):



6 - Buffer triestado (B)

2.1.2 Defectos eléctricos

2.1.2.1 Qué es un defecto eléctrico

Cuando el estado en que se encuentra una red se encuentra condicionado por más factores que el valor de sus buffers de entrada, se dice que existe un defecto eléctrico en dicha red.

2.1.2.2 Tipos de defectos

Los defectos ocasionan que el comportamiento de la red sea inesperado, sin embargo, los defectos se pueden catalogar según su comportamiento en dos grandes grupos, defectos deterministas y defectos no deterministas.

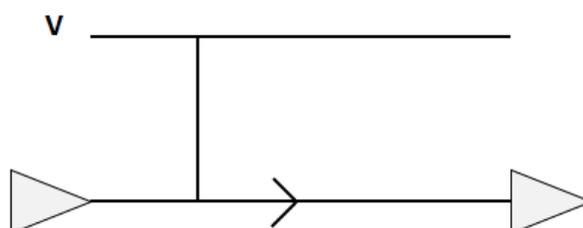
Los defectos deterministas, como indica su nombre, causan un comportamiento conocido sobre la red que actúan, sin embargo, los no deterministas causan un comportamiento caótico.

El tipo de fallos deterministas que pueden ocurrir en una tarjeta de circuito impreso son los siguientes:

- **En una red:**

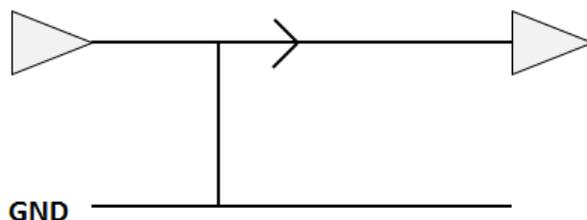
Los defectos que afectan a una red de manera aislada se denominan de tipo *stuck* y se clasifican en:

- *Stuck-at-one*: Estos defectos producen siempre un valor de 1 lógico. No implica que la red esté necesariamente conectada a una toma con la tensión de alimentación.[4]



7 - Ejemplo de defecto *Stuck-at-one*

- *Stuck-at-zero*: Estos defectos originan en la red un efecto tal que el valor lógico de la misma se corresponde con un 0 lógico. No precisa necesariamente de conexión directa a tierra.[4]



8 - Ejemplo de red Stuck-at-zero

- *Open fault* o *Stuck-open*: Ocurre en aquellos casos en los que la entrada no se encuentra conectada al buffer de salida. Se comportan como *stuck-at-one* o *stuck-at-zero* según la tecnología del buffer que lee la señal.

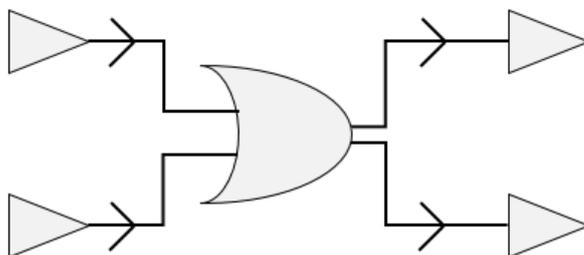


9 - Ejemplo de red con defecto open fault

- **Entre varias redes:**

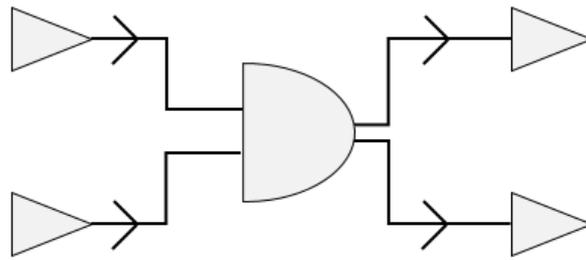
Los defectos que afectan a varias redes se denominan de tipo *short*, y se clasifican en:

- *OR-type shorts*: Estos fallos generan en los buffers de salida de las redes afectadas un resultado equivalente a una puerta lógica OR entre las redes afectadas. Se comportan de tal manera que domina el 0 lógico.



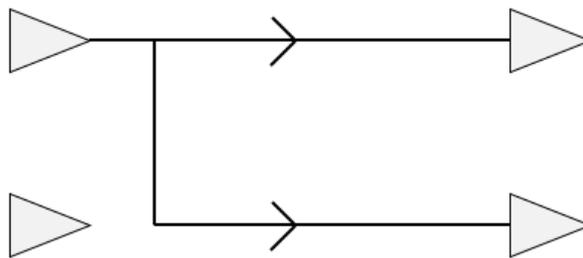
10 - Comportamiento de un defecto OR-type short

- *AND-type shorts*: El resultado a la salida de la red está dominado por un 1 lógico. El efecto del corto es idéntico a incluir una puerta lógica AND entre las redes afectadas.



11 - Comportamiento de un defecto AND-type short

- o Strong-driven shorts: En este caso, los buffers de salida están dominados por un solo input, el que tenga mayor influencia de todos aquellos a los que afecte el corto.



12 - Comportamiento de un defecto Strong-driven short

En el caso de fallos entre varias redes, para un cortocircuito concreto entre unas redes determinadas, solo un tipo de comportamiento de los aquí presentados es posible. El comportamiento que tendrá lugar depende de la tecnología de los diferentes buffers.[3]

2.1.3 Estándar IEEE 1149.1

En 1980 el *Joint Test Action Group* (JTAG) desarrolló una especificación para realizar un test periférico a los circuitos integrados. En 1990 se estandarizó esta especificación como el estándar IEEE 1149.1-1990. Posteriormente se realizaron revisiones como la de 1993, resultando en el IEEE 1149.1a y se añadieron suplementos al mismo como el de 1994, que incluyó un fichero con la descripción del lenguaje *boundary scan*.

Desde entonces este estándar ha sido adoptado por los fabricantes mayoritarios de electrónica [5]. Se encuentra especialmente presente en productos de telecomunicaciones, sistemas de defensa, ordenadores y aeronáutica.

Pese a estar diseñado originalmente para realizar la prueba de módulos de circuitos integrados, también es ampliamente usado para depurar sistemas embebidos, pues permite leer y escribir los valores periféricos de los circuitos. Algunos fabricantes incluyen incluso instrucciones propietarias que permiten acceder a registros internos de los propios chips. Otro de los usos que tiene es la programación de sistemas con lógica programable como FPGAs y microcontroladores.

El sistema de *boundary-scan* definido por el estándar permite el control y la observación de los pines del circuito integrado. Esto permite acceder directamente a las redes que se encuentra conectado dicho integrado. Al poderse acceder así directamente a las redes anexas al circuito, se reducen los tiempos empleados para realizar diversos test, a la par que se aumenta la cobertura y capacidad de diagnóstico de los mismos, lo que reduce el coste al momento de realizar las diferentes pruebas.

El estándar marca una serie de componentes que ha de incluir todo dispositivo compatible con la normativa. Entre ellos están el conjunto de señales admitidas, el controlador del puerto de acceso para test, los registros empleados y las celdas para *boundary-scan*.

El estándar analizado en este trabajo fin de grado es el revisado en 2001[1].

2.1.3.1 Señales empleadas

El estándar define un total de cinco señales, de las cuales una es opcional:

- **TCK** (Test Clock): Obligatoria. La señal de reloj que gobierna toda la lógica interna de los circuitos relativa al test del *boundary-scan*. Cuando esta señal se detenga en 0, toda la lógica del sistema ha de mantener su estado actual. Se permite que ocurra lo mismo cuando la señal se mantenga como un 1 lógico.
- **TMS** (Test Mode Select): Obligatoria. Esta señal es la que controla el puerto de acceso para test. Su lectura se realiza en el flanco de subida de TCK, por lo que se espera que el controlador maestro modifique el valor en los flancos de bajada.
- **TDI** (Test Data Input): Obligatoria. Es el valor de entrada para el test de instrucciones o datos. El valor de TDI se realiza en el flanco de subida de TCK, por lo que se espera que el controlador maestro genere el próximo valor de TDI en los flancos de subida.
- **TDO** (Test Data Output): Obligatoria. Es el valor recibido a la salida del test de instrucciones o datos. El cambio de la señal ocurre en el flanco de bajada.
- **TRST** (Test Reset): Opcional. Si el sistema no revierte por defecto al estado de *reset* con el encendido, es obligatorio incluir soporte para esta señal.

La conexión de las señales entre los diversos componentes que formen la cadena de test ha de ser la siguiente:

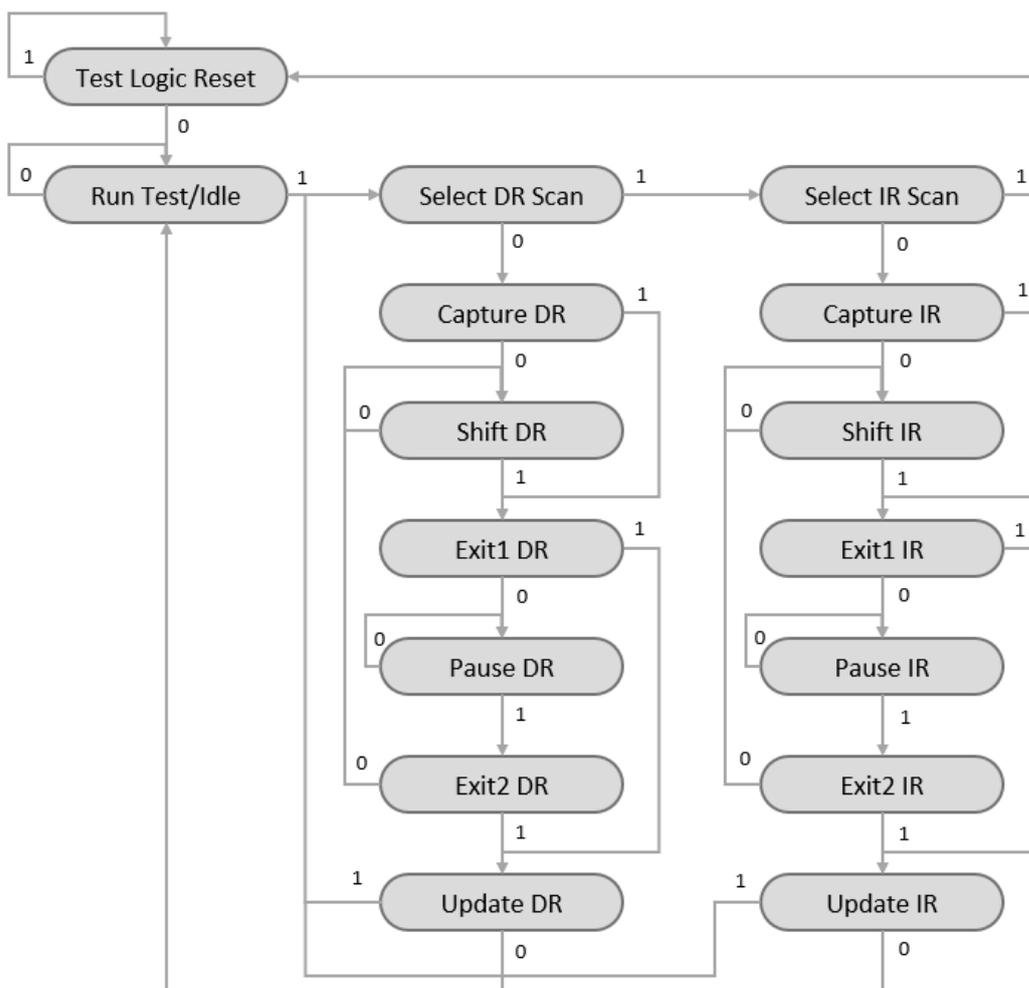
- TCK: Conectado en paralelo a todos los dispositivos
- TMS: Conectado en paralelo a todos los dispositivos
- TDI: Conectado al TDO del dispositivo previo, o en su defecto al controlador maestro.
- TDO: Conectado al TDI del dispositivo siguiente, en su defecto al controlador maestro.
- TRST: Conectado en paralelo a todos los dispositivos que admitan la señal.

2.1.3.2 Puerto de acceso para test

El puerto de acceso para test o *Test Access Port* (TAP) cuenta con las señales indicadas previamente como interfaz. Estas señales gobiernan un controlador, encargado de seleccionar el registro sobre el que se realiza la etapa del test, y que permite que todas las señales de entrada y salida puedan ser controladas y observadas[2].

El controlador del puerto de acceso para test es una máquina de estados finitos. Cuenta con 16 estados, 7 dedicados al registro de instrucciones, 7 dedicados a los registros de datos, 1 para el reinicio de la lógica, y otro en el que el controlador se pausa.

El diagrama de estados se muestra a continuación:



13 - Máquina de estados del puerto de acceso para test

La máquina de estados está controlada por la señal TMS, cuyo valor marca cual es el próximo estado que tomará el controlador con el siguiente flanco de subida de TCK.

Las operaciones indicadas con DR se refieren al registro de datos seleccionado previamente por la instrucción cargada en el controlador correspondiente.

A continuación se explica brevemente cada uno de los 16 estados.

- **Test Logic Reset:** Se deshabilita toda la lógica, permitiendo un funcionamiento normal del circuito integrado. Sea cual sea el estado en que se encuentre la máquina de estados, si se mantiene TMS a 1 durante 5 ciclos, se llega a este estado.
- **Run Test / Idle:** En este estado, la lógica del integrado solo se mantiene activa si se han cargado ciertas instrucciones como SELFTEST. En el resto de casos, la lógica del integrado mantiene su valor previo.
- **Select DR Scan:** Estado de transición, permite seleccionar si se activa la cadena de datos o si se pasa al estado Select IR Scan.
- **Select IR Scan:** Estado de transición, controla si se activa la cadena de instrucciones o si se vuelve al estado Test Logic Reset.
- **Capture IR:** En este estado se carga en el registro de instrucciones el código INSTRUCTION_CAPTURE de cada integrado, un patrón predefinido en los integrados cuyos últimos valores ha de ser "01".
- **Shift IR:** Este estado conecta el registro de instrucciones entre TDI y TDO del integrado, de tal forma que en cada flanco de subida de TCK se realiza un desplazamiento del último bit hacia TDO mientras que el entrante por TDI ocupa el primer bit.
- **Exit1 IR:** Estado de transición, controla si se pasa al estado Pause IR o Update IR.
- **Pause IR:** Este estado permite detener temporalmente el desplazamiento del registro de instrucciones. Útil en equipos de test automático (ATE).
- **Exit2 IR:** Estado de transición que permite elegir si se pasa al estado Shift IR o Update IR.

- **Update IR:** En este estado, la instrucción presente en el registro de instrucciones se carga en el integrado en el flanco de bajada de TCK, convirtiéndose en la instrucción que pasaría a ejecutar el integrado.
- **Capture DR:** Estado que carga los datos correspondientes a la instrucción activa en el registro correspondiente.
- **Shift DR:** Este estado conecta el registro de datos seleccionado por la instrucción activa entre TDI y TDO del integrado, de tal forma que en cada flanco de subida de TCK se realiza un desplazamiento del último bit hacia TDO mientras que el entrante por TDI ocupa el primer bit.
- **Exit1 DR:** Estado de transición, controla si se pasa al estado Pause DR o Update DR.
- **Pause DR:** Este estado permite detener temporalmente el desplazamiento del registro de datos correspondiente. Útil en equipos de test automático (ATE).
- **Exit2 DR:** Estado de transición que permite elegir si se pasa al estado Shift DR o Update DR.
- **Update DR:** En este estado, la cadena presente en el registro de datos se carga en el integrado o en la salida del mismo en el flanco de bajada de TCK.

2.1.3.3 El registro de instrucciones

Este registro indica al controlador de puerto de acceso para test qué tipo de test realizar. Para ello se sirve de las denominadas instrucciones. Cualquier dispositivo que implemente el estándar ha de incluir obligatoriamente las siguientes instrucciones:

- **EXTEST:** Permite realizar un test de interconexión entre dispositivos, o un test de pistas. Indica que el registro de datos a conectar entre TDI y TDO es el registro *boundary-scan*.
- **BYPASS:** Reduce la longitud del registro de datos a un solo bit, esto permite reducir la longitud de la cadena de test a la vez que se permite el funcionamiento normal del dispositivo. Esta instrucción ha de estar formada exclusivamente por valores 1 lógico {1, 1, 1... 1, 1}. Esta

instrucción conecta el registro de bypass entre TDI y TDO. Este registro se carga por defecto con un 0 lógico al entrar el controlador en el estado Capture DR.

- **SAMPLE/PRELOAD:** Permite que el dispositivo mantenga su funcionamiento normal y conecta el registro de *boundary-scan* entre TDI y TDO, permitiendo así tomar muestras del estado en que se encuentran las conexiones del integrado. También permite precargar el registro previamente a cargar una instrucción EXTEST.

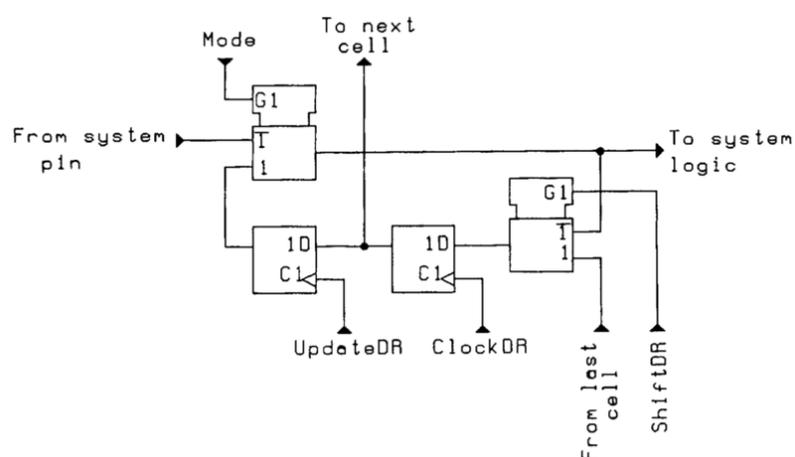
2.1.3.4 El registro *boundary-scan*

El registro de *boundary-scan*, se basa en un registro con desplazamiento, cuyas celdas varían acorde a los requisitos del componente. Las celdas son distintas según se trate de un pin de entrada, salida, triestado o bidireccional.

Este registro permite realizar un test de interconexiones en la placa, detectando los diferentes defectos eléctricos que pueden existir en ella.

Las diferentes celdas están conectadas en serie, lo que permite realizar el desplazamiento de sus valores manteniendo el orden de la cadena *boundary-scan*. A la vez se encuentran conectadas en paralelo a las entradas y salidas tanto de la lógica interna como de las patillas del chip, lo que permite capturar y actualizar los valores de todas las celdas al mismo tiempo.

A continuación se muestra un ejemplo de celda *boundary-scan*:



14 - Muestra de una celda *boundary scan*. Celda BC_2 de [1]

En un funcionamiento normal del dispositivo, este registro no juega ningún papel en el dispositivo, pues actúa como un puente directo entre las entradas y salidas que conecta.

En el caso de estar ejecutándose la instrucción EXTEST expuesta previamente, este registro actuaría de la siguiente forma:

- En el estado CAPTURE DR, el valor a la entrada de las celdas es almacenado en la propia celda.
- En el estado SHIFT DR, el valor de la celda previa es almacenado a la par que el valor previo es enviado a la celda posterior en la cadena.
- En el estado UPDATE DR el valor almacenado en las celdas se transfiere a la salida de las mismas.

2.1.3.5 Los archivos BSDL

No fue hasta 1994, tras la aparición del estándar IEEE 1149.1 en 1990 y las modificaciones realizadas en 1993, cuando se promulgó la primera definición de BSDL. En 2001 se realizó una revisión del mismo, que es la versión aquí presentada.

Todo fabricante de dispositivos compatibles con este estándar, ha de facilitar un fichero denominado *Boundary Scan Description Language* (BSDL) en el que se indican las siguientes características del componente:

- Longitud y estructura del registro boundary-scan.
- Existencia del pin opcional TRST.
- Localización física de los pines correspondientes a las señales del puerto de acceso para test.
- Códigos binarios de las instrucciones implementadas.
- Código de identificación del dispositivo

Otra serie de características que es opcional incluir son las siguientes:

- Diagrama de estados del controlador del puerto de acceso para test.
- Registro de bypass

- Longitud del registro que proporciona el código de identificación del dispositivo.
- Disposición de las instrucciones BYPASS, SAMPLE, PRELOAD y EXTEST.
- Funcionamiento de otras instrucciones.

A continuación se mostrará un ejemplo de las diversas partes con las que cuenta en fichero BSDL.

Descripción de los puertos lógicos. Esto incluye la denominación de los puertos, así como el tipo de pin del puerto correspondiente. En la siguiente imagen podemos ver que el puerto TCK es de tipo entrada, el puerto TDO es de tipo salida y el puerto VSS es de alimentación. Se aprecia que todos ellos tienen una longitud de un solo bit.

```
TCK      :      in      bit;
TDO      :      out     bit;
VSS      :      linkage bit;
```

15 - Descripción de puertos lógicos en un fichero BSDL

Seguidamente se indica la distribución física de los pines, es decir, el orden físico que ocupan en cuanto a las conexiones del circuito integrado. En la siguiente imagen se ve cómo el puerto TCK corresponde a la patilla 17, seguido por el puerto TDO y el VSS2 que se corresponde con la 19.

```
" TCK      :      17 ," &
" TDO      :      18 ," &
" VSS2     :      19 ," &
```

16 - Distribución física de los pines en un fichero BSDL

Otra parte presente en estos ficheros es la que se corresponde a la identificación de las señales que gobiernan el TAP. Es de especial importancia la última línea, la correspondiente a TAP_SCAN_CLOCK, pues nos indica la máxima frecuencia a la que es capaz de funcionar el controlador TAP. En este caso es de 10 MHz:

```
attribute TAP_SCAN_IN of TDI : signal is true;
attribute TAP_SCAN_MODE of TMS : signal is true;
attribute TAP_SCAN_OUT of TDO : signal is true;
attribute TAP_SCAN_CLOCK of TCK : signal is (10.00e+06,BOTH);
```

17 - Definición de la interfaz del TAP en un fichero BSDL

Un punto clave es la longitud del registro de instrucciones, como se puede apreciar, en este ejemplo la longitud es de 5 bits:

```
attribute INSTRUCTION_LENGTH of PIC32MX110F016B : entity is 5;
```

18 - Longitud del registro de instrucciones en un fichero BSDL

Posteriormente se indican los códigos binarios correspondientes a las instrucciones implementadas en el controlador TAP. Como podemos ver, en este caso la instrucción EXTEST tiene asignado el código "00110":

```
attribute INSTRUCTION_OPCODE of PIC32MX110F016B : entity is
    "extest (00110)," &
    "bypass (11111)," &
    "sample (00010)," &
```

19 - Códigos binarios de instrucciones en un fichero BSDL

Es de especial importancia el código que se carga por defecto en el registro de instrucciones al pasar el controlador por el estado CAPTURE IR, en este caso podemos ver que se cargaría el valor "00001":

```
attribute INSTRUCTION_CAPTURE of PIC32MX110F016B : entity is "00001";
```

20 - Código cargado en CAPTURE IR en un fichero BSDL

Finalmente se indican la descripción del registro *boundary-scan*. Este incluye la posición de la celda en el registro, siendo 0 la celda más próxima a TDO, el bit menos significativo. También se indica el tipo de celda de acuerdo al estándar, la denominación del puerto, y la función de la celda. En el caso de celdas controladas, también se indica qué celda la gobierna, el valor que necesita tener dicha celda para deshabilitar la controlada, y el efecto valor que tendría la celda en caso de que eso ocurriera. Aquí se presenta una muestra, en el que el puerto RB4 cuenta con tres celdas. La celda 6, RB4, es de entrada. La celda 7, RB4, es de salida, y está controlada por la celda 8, de tal manera que cuando la celda 8 tome el valor lógico 0, la celda 7 entrará en estado de alta impedancia:

```
"6 ( BC_4, RB4, input, X)," &
"7 ( BC_1, RB4, output3, X, 8, 0, Z)," &
"8 ( BC_2, *, control, 0)," &
```

21 - Descripción del registro *boundary-scan* en un fichero BSDL

2.1.4 Algoritmos

2.1.4.1 Consideraciones sobre algoritmos para test de pistas

Entre otras ventajas, el sistema de *boundary-scan* permite que equipos de test automáticos (ATE) de bajo coste puedan verificar las conexiones de las placas. Un simple ordenador, sin componentes especializados, es capaz de realizar el test, sin embargo, se encontrará limitado por características como la capacidad de procesamiento y la velocidad de acceso al disco. Se ha de valorar la capacidad del ATE de producir los vectores de test en tiempo real, según vayan siendo necesarios, la velocidad a la que es capaz de enviar los mismos, y el procesamiento que hace con los vectores que recibe.

Estas características determinan el tipo de algoritmo que debiera emplear el dispositivo para lograr un funcionamiento óptimo. Algunos algoritmos requieren una gran capacidad de procesamiento para la generación de los vectores, lo que puede dificultar el diagnóstico. Otros algoritmos de identificación pueden precisar una gran cantidad de memoria para almacenar todas las respuestas recibidas, y en otros casos se tiene que contar con complejos sistemas para procesar las respuestas recibidas.

Como se puede deducir de lo anteriormente expuesto, hay algoritmos que requieren una elevada capacidad de procesamiento, ya sea en la generación de los vectores de test o en el tratamiento de los mismos. Estos algoritmos, típicamente ofrecen menores tiempos de test a la par que precisan de mayor cantidad de recursos.

Independientemente del algoritmo empleado, se ha de tener en cuenta que algunos ATE proveen un resultado del test comprimido, mientras que otros detienen el test en cuanto detectan cierta cantidad de fallos[2]. En estos casos, la información de la que se dispone es incompleta, lo que dificulta el diagnóstico correcto.

Los algoritmos empleados en este tipo de test se pueden clasificar en dos grandes grupos, los algoritmos de una etapa y los algoritmos adaptativos. Los algoritmos de una etapa, emplean un solo algoritmo para detectar los fallos,

mientras que los algoritmos adaptativos ejecutan un primer algoritmo para detectar la presencia de fallos, seguido por un segundo algoritmo que localiza e identifica los fallos presentes.

Los algoritmos de una sola etapa, de forma general conllevan mayor tiempo de test que los adaptativos, pues estos últimos realizan un primer test de menor tiempo que detecta la presencia de fallos. Es solo en el caso de que existan fallos cuando este tipo de test podría llegar incluso a superar en duración a los de una etapa si la presencia de fallos es elevada.

En nuestro caso, los algoritmos que se valorarán son los de una sola etapa, pues las FPGAs precisan estar programadas previamente, y la segunda parte de los algoritmos adaptativos varía según la cantidad y localización de los fallos identificados en el primer test, lo que implica emplear distinta lógica en esta segunda parte.

Todos los algoritmos emplean los denominados vectores de test:

Red	Vectores de test paralelos				Vectores de test secuenciales
	V ₁	V ₂	V ₃	V ₄	
n ₁	0	0	0	1	V ₁
n ₂	0	0	1	0	V ₂
n ₃	0	0	1	1	V ₃
n ₄	0	1	0	0	V ₄
n ₅	0	1	0	1	V ₅
n ₆	0	1	1	0	V ₆
n ₇	0	1	1	1	V ₇
n ₈	1	0	0	0	V ₈

1 - Ejemplo de vectores de test

Los vectores de test paralelos son el total de vectores de test que se envían al registro *boundary-scan*. En este caso hay cuatro vectores de test paralelos.

Los vectores secuenciales son el total de estados que se envían a una celda concreta del registro *boundary-scan*. En este caso hay ocho vectores de test secuenciales.

El análisis de las respuestas también se realiza en base a los vectores recibidos:

Red	Vectores de respuesta paralelos				Vectores de respuesta secuenciales
	v ₁	v ₂	v ₃	v ₄	
n ₁	0	0	0	1	v ₁
n ₂	0	0	1	0	v ₂
n ₃	0	0	1	1	v ₃
n ₄	0	1	0	0	v ₄
n ₅	0	1	0	1	v ₅
n ₆	0	1	1	0	v ₆
n ₇	0	1	1	1	v ₇
n ₈	1	0	0	0	v ₈

2 - Muestra de vectores de respuesta recibidos

Los vectores de respuesta paralelos son los estados que tiene cada una de las celdas de captura de la red para los vectores de test paralelos enviados previamente. Se observa que coincide la cantidad de vectores de respuesta paralelos, así como la cantidad de elementos con los vectores de test enviados.

Los vectores de respuesta secuenciales son los estados por los que ha pasado una de las celdas de salida del registro *boundary-scan*. Se observa que coincide la cantidad de estados recibidos con la cantidad de estados enviados en los vectores de test secuenciales.

2.1.4.2 Diagnóstico de los fallos

Cuando se realiza el tratamiento de los vectores respuesta, se puede conocer qué tipo de fallo existe y dónde está localizado el mismo, lo que se denomina identificar el defecto. En el caso en que sea imposible determinar la localización del mismo, o que solo se pueda conocer dónde hay un fallo, pero no se pueda identificar el mismo, se habla de detección del defecto.

Realizar una buena identificación de los fallos es importante para poder reducir el tiempo empleado para reparar una placa con defectos.

La capacidad de diagnóstico de un algoritmo no guarda relación con el tiempo empleado por el mismo ni por la capacidad de procesamiento que requiera. No obstante, ciertos algoritmos, realizan un tratamiento de las señales o un envío de vectores que pueden dar lugar a ciertos fallos en el diagnóstico, los denominados síndromes[6].

- Aliasing syndrome – Síndrome de solapamiento

Aparece cuando los vectores de respuesta secuenciales de varias redes cortocircuitadas entre ellas son idénticos al de una red sin fallos. En este caso no se puede saber si la respuesta de la red libre de fallos está causada por el cortocircuito.

Red	Vectores de test paralelos				Vectores de test secuenciales
	V ₁	V ₂	V ₃	V ₄	
n ₁	0	0	0	1	V ₁
n ₂	0	0	1	0	V ₂
n ₃	0	0	1	1	V ₃

3 - Ejemplo de vectores de test previos a un aliasing syndrome

Red	Vectores de respuesta paralelos				Vectores de respuesta secuenciales
	V ₁	V ₂	V ₃	V ₄	
n ₁	0	0	1	1	V ₁
n ₂	0	0	1	1	V ₂
n ₃	0	0	1	1	V ₃

4 - Muestra de vectores de respuesta posteriores a un aliasing syndrome

Como se puede ver, en este caso los vectores de respuesta secuenciales son idénticos en las tres redes. Los dos primeros vectores de respuesta se deben a cortocircuito de tipo OR entre ellos. El tercer vector se encuentra libre de fallos, no obstante, no se puede demostrar que su respuesta no sea la provocada por el cortocircuito entre las otras dos pistas. Ocurre el denominado síndrome de solapamiento, una respuesta con defecto es idéntica a una respuesta sin fallo.

- Confounding syndrome – Síndrome de confusión

Aparece cuando los vectores de respuesta secuenciales de un cortocircuito son idénticos a los vectores de respuesta secuencias de otro cortocircuito distinto.

Red	Vectores de test paralelos				Vectores de test secuenciales
	v ₁	v ₂	v ₃	v ₄	
n ₁	0	0	0	1	v ₁
n ₂	0	0	1	0	v ₂
n ₃	0	0	1	1	v ₃
n ₄	0	1	0	0	v ₄
n ₅	0	1	0	1	v ₅

5 - Ejemplo de vectores de test previos a un confounding syndrome

Red	Vectores de respuesta paralelos				Vectores de respuesta secuenciales
	v ₁	v ₂	v ₃	v ₄	
n ₁	0	0	0	1	v ₁
n ₂	0	1	1	1	v ₂
n ₃	0	1	1	1	v ₃
n ₄	0	1	1	1	v ₄
n ₅	0	1	1	1	v ₅

6 - Ejemplo de vectores de respuesta posteriores a un confounding syndrome

Tal y como se observa en la tabla previa, los vectores de respuesta secuenciales de las redes 2, 3, 4 y 5 tienen idéntico valor. Las redes 2 y 5 se encuentran cortocircuitadas entre sí. Las redes 3 y 4 también se encuentran cortocircuitadas entre ellas mismas, sin embargo, la respuesta que producen es idéntica a la del otro cortocircuito. Se ha de notar que la respuesta también coincide con un cortocircuito tipo OR de las cuatro pistas.

Ocurre el denominado síndrome de confusión, no se puede conocer a qué cortocircuito se debe la respuesta recibida.

Hay que destacar que es posible que ambos síndromes ocurran a la par.

2.1.4.3 Algoritmos generadores de vectores de test

Los algoritmos aquí presentados son algoritmos de una sola etapa, no se incluyen algoritmos adaptativos por la dificultad que supondría implementar los mismos en una FPGA.

- **Counting Sequence Algorithm**

Una de las primeras propuestas para realizar este tipo de test es la realizada por W. K. Kautz[7]. Este algoritmo asigna un vector de test secuencial único a cada red, comenzando por un vector compuesto enteramente por ceros, lo que permite la detección óptima de cortocircuitos.

Red	Vectores de test paralelos				Vectores de test secuenciales
	V ₁	V ₂	V ₃	V ₄	
n ₁	0	0	0	0	V ₁
n ₂	0	0	0	1	V ₂
n ₃	0	0	1	0	V ₃
n ₄	0	0	1	1	V ₄
n ₅	0	1	0	0	V ₅
n ₆	0	1	0	1	V ₆
n ₇	0	1	1	0	V ₇
n ₈	0	1	1	1	V ₈
n ₉	1	0	0	0	V ₉
n ₁₀	1	0	0	1	V ₁₀

7 - Muestra del Counting Sequence Algorithm

Si la red está libre de fallos, cada respuesta será única, y en caso de cortocircuito, las redes involucradas tendrán el mismo comportamiento[2].

No permite detectar fallos de tipo *stuck-at*[4], [7] debido a que debe existir siempre un vector compuesto por todo ceros e incluso puede existir otro compuesto por todo unos.

Este algoritmo tiene un gran inconveniente, pues presenta aliasing syndrome y confounding syndrome[4].

- **Modified Counting Sequence Algorithm**

Este algoritmo se basa en el *Counting Sequence Algorithm*, lo que hace es prescindir de los vectores compuestos enteramente de ceros y unos. Fue propuesto por P. Goel[8].

Red	Vectores de test paralelos				Vectores de test secuenciales
	v ₁	v ₂	v ₃	v ₄	
n ₁	0	0	0	1	v ₁
n ₂	0	0	1	0	v ₂
n ₃	0	0	1	1	v ₃
n ₄	0	1	0	0	v ₄
n ₅	0	1	0	1	v ₅
n ₆	0	1	1	0	v ₆
n ₇	0	1	1	1	v ₇
n ₈	1	0	0	0	v ₈
n ₉	1	0	0	1	v ₉
n ₁₀	1	0	1	0	v ₁₀

8 - Muestra del Modified Counting Sequence Algorithm

Este algoritmo resuelve el problema en cuanto a la detección de los fallos tipo stuck-at[6] al incluir al menos un uno y un cero en cada red. Sin embargo, al contar los vectores secuenciales de respuesta con varios unos, los síndromes de superposición y confusión todavía pueden aparecer.

- **True/Complement Test and Diagnosis Algorithm**

Este algoritmo fue propuesto por P. T. Wagner[9] para eliminar el síndrome de superposición, presente en los algoritmos previos. El total de vectores de test empleados es el doble a los otros algoritmos. Para obtener el segundo par de vectores, se complementan los primeros.

Red	Vectores de verdad				Vectores de test secuenciales	Vectores complemento				Vectores de test secuenciales
	V1	V2	V3	V4		V1	V2	V3	V4	
n1	0	0	0	1	V1	1	1	1	0	V11
n2	0	0	1	0	V2	1	1	0	1	V12
n3	0	0	1	1	V3	1	1	0	0	V13
n4	0	1	0	0	V4	1	0	1	1	V14
n5	0	1	0	1	V5	1	0	1	0	V15
n6	0	1	1	0	V6	1	0	0	1	V16
n7	0	1	1	1	V7	1	0	0	0	V17
n8	1	0	0	0	V8	0	1	1	1	V18
n9	1	0	0	1	V9	0	1	1	0	V19
n10	1	0	1	0	V10	0	1	0	1	V20

9 - Muestra del True/Complement Test and Diagnosis Algorithm

Este algoritmo permite detectar los fallos *stuck-at* así como cortocircuitos. Como se ha explicado, el fin de esta propuesta es eliminar los *aliasing syndromes*, sin embargo los *confounding syndromes* todavía pueden ocurrir[4].

- **Walking One's Algorithm**

Este algoritmo, permite diagnosticar todos los tipos de defectos eléctricos presentes en una tarjeta de circuito impreso[6]. Este algoritmo solo permite la existencia de un uno en cada vector de test paralelo, de tal manera que el resto son ceros. Para cada vector de test paralelo la posición del 1 ha de ser distinta, por lo que se enviarán tantos vectores como redes haya en el circuito.

Red	Vectores de test paralelos										Vectores de test secuenciales
	V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀	
n ₁	0	0	0	0	0	0	0	0	0	1	V ₁
n ₂	0	0	0	0	0	0	0	0	1	0	V ₂
n ₃	0	0	0	0	0	0	0	1	0	0	V ₃
n ₄	0	0	0	0	0	0	1	0	0	0	V ₄
n ₅	0	0	0	0	0	1	0	0	0	0	V ₅
n ₆	0	0	0	0	1	0	0	0	0	0	V ₆
n ₇	0	0	0	1	0	0	0	0	0	0	V ₇
n ₈	0	0	1	0	0	0	0	0	0	0	V ₈
n ₉	0	1	0	0	0	0	0	0	0	0	V ₉
n ₁₀	1	0	0	0	0	0	0	0	0	0	V ₁₀

10 - Muestra de Walking One's Algorithm

Existen variaciones de este algoritmo que incluyen una compresión de la respuesta, este caso no será estudiado.

Este algoritmo permite detectar tanto los fallos de tipo *stuck-at* como los cortocircuitos. Pese a requerir mayor cantidad de vectores de test que los algoritmos previos, evita la aparición de los síndromes de superposición y confusión[10] pues el patrón permite identificar claramente las redes que entran en conflicto en cada defecto eléctrico.

Puesto que los vectores de test propuestos por este algoritmo son diagonalmente independientes[2], si se emplean los vectores complementarios, al estilo del “*True/Complement Test and Diagnosis Algorithm*” se garantiza un diagnóstico completo. Este se denomina Walking One/Walking Zero Algorithm.

Se trata del único algoritmo de una etapa capaz de realizar un diagnóstico completo.

2.1.4.4 Comparación de las principales alternativas

Algoritmo	Vectores de test paralelos	Detección		Síndromes	
		Stuck-at	Cortocircuitos	Superposición	Confusión
Counting Sequence Algorithm	$\lceil \log(N) \rceil$	NO	SI	SI	SI
Modified Counting Sequence Algorithm	$\lceil \log(N + 2) \rceil$	SI	SI	SI	SI
True/Complement Test and Diagnosis Algorithm	$2 * \lceil \log(N) \rceil$	SI	SI	NO	SI
Walking One's Algorithm	N	SI	SI	NO	NO
Walking One/Walking Zero Algorithm	$2 * N$	SI	SI	NO	NO

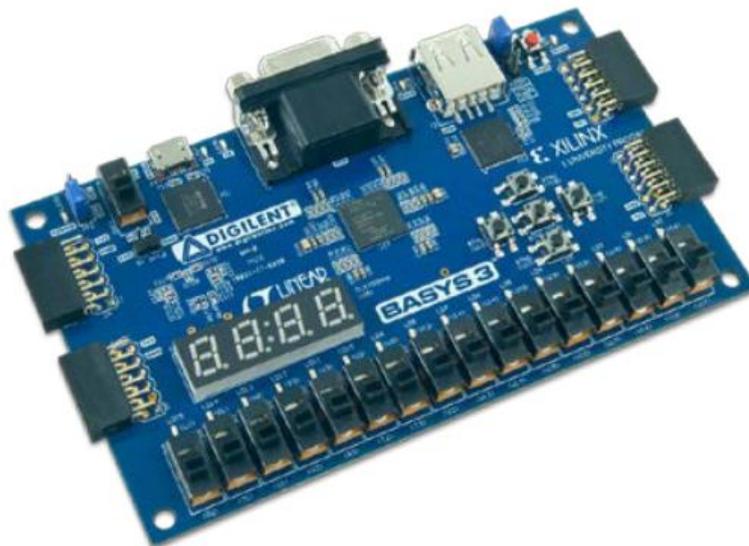
11 - Comparación de los distintos algoritmos propuestos

En el caso de los algoritmos “*Counting Sequence Algorithm*” y “*Modified Counting Sequence Algorithm*” se ha de tener en cuenta que no detectan los cortocircuitos de tipo AND.

El *Walking One/Walking Zero Algorithm* permite identificar correctamente todos los tipos de cortocircuitos, pues el *Walking One* es incapaz de detectar los cortocircuitos de tipo AND mientras que el *Walking Zero* no puede identificar los cortocircuitos de tipo OR. Al usar ambas versiones del mismo algoritmo se logra un diagnóstico completo.

2.2 Plataforma de desarrollo

Debido a la disponibilidad en el laboratorio, la plataforma empleada en el desarrollo será la tarjeta Basys 3 de Digilent[11].



22 - Basys 3 [12]

Esta tarjeta emplea como unidad central una FPGA de Xilinx de su agama Artix 7. Concretamente el modelo XC7A35T-1CPG236C[13].

Esta FPGA ofrece:

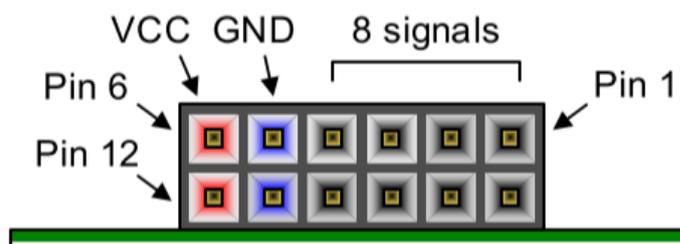
- 33280 celdas lógicas en 5200 slices con 6-input LUTs y 8 flip-flops.
- 1800 Kbits de RAM.
- Control sobre cinco relojes con PLL.
- 90 slices para DSP.
- Reloj interno con frecuencia superior a los 450 MHz.
- Conversión analógica-digital (XADC).

La Basys 3 incorpora otras características, como las siguientes:

- 16 interruptores
- 16 LEDs
- 4 displays de 7 segmentos.
- Conector VGA de 12 bits.
- Puerto USB-JTAG para programar y comunicar con la FPGA.

- 3 conectores PMOD
- Puente USB-UART
- Soporte USB para ratón, teclado y dispositivos de memoria.
- 5 botones.
- Conector PMOD para señales analógicas XADC.
- Serial Flash.

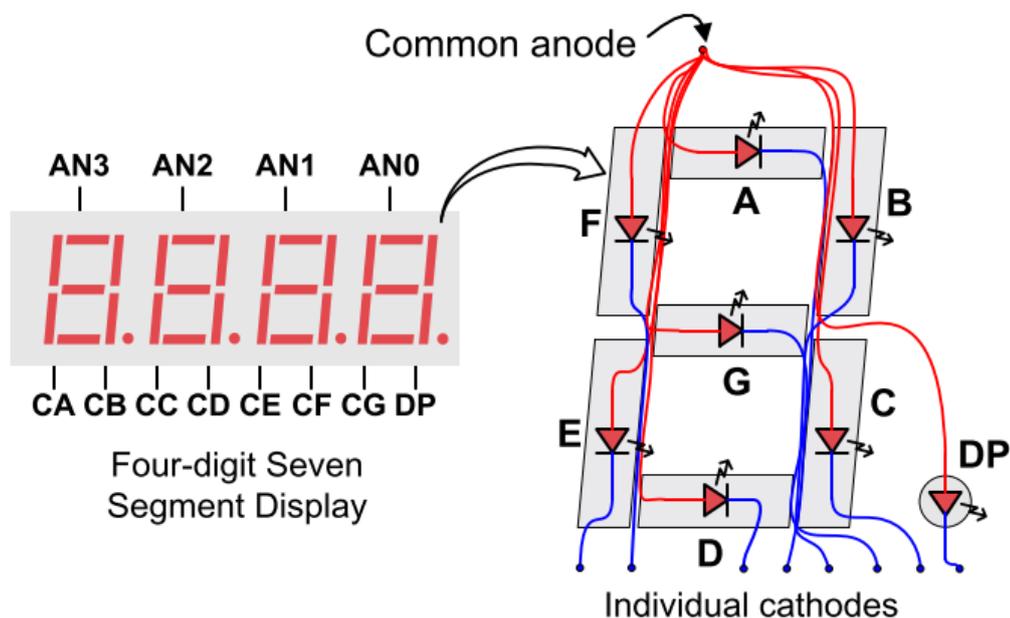
Entre todas ellas, destacaremos el puerto PMOD, pues es el que se empleará para conectar la FPGA al circuito generador de defectos eléctricos.



23 - Puerto PMOD en [12]

Este puerto nos permite alimentar los circuitos de dicha placa, así como comunicarnos con ella.

Otro elemento que cobra especial importancia es el display de 4 dígitos con 7 segmentos:



24 - Display 4 dígitos de 7 segmentos en [12]

Este display se empleará para mostrar el resultado del test, así como las diversas etapas por las que pasa.

Como se ha mencionado previamente, el corazón de la Basys 3 es un FPGA.

La primera FPGA comercial fue inventada por Ross Freeman y Bernard Vonderschmitt, co-fundadores de Xilinx en 1985, esta FPGA fue comercializada como XC2064[14].

Las FPGA surgen como respuesta ante la necesidad de disponer de dispositivos lógicos programables (PLD) con un rendimiento similar a los circuitos diseñados para aplicaciones concretas, los *Application-Specific Integrated Circuit* (ASIC). Estos últimos son diseñados a medida según los requisitos especificados, por lo que su coste es muy elevado.

Gracias a que son programables múltiples veces, sin que esto afecte a su buen rendimiento, las FPGAs se emplean a menudo en prototipos previos al diseño de ASICs, e incluso se comercializan ya programadas sustituyendo a los mismos.

Las FPGAs están compuestas por bloques de puertas lógicas configurables por medio de un lenguaje de descripción hardware (HDL). Este lenguaje puede ser cualquiera de los siguientes:

- VHDL
- ABEL
- Verilog

En el desarrollo de este trabajo se considerará el uso de VHDL. Se trata de un lenguaje definido por el IEEE, empleado para describir circuitos digitales. El nombre aparece como resultado de mezclar *Very High Speed Integrated Circuit* (VHSIC) y *Hardware Description Language* (HDL). Aunque puede emplearse para describir cualquier circuito digital, se emplea principalmente para la programación de dispositivos lógicos programables (PLD), FPGAs, ASICs y similares.

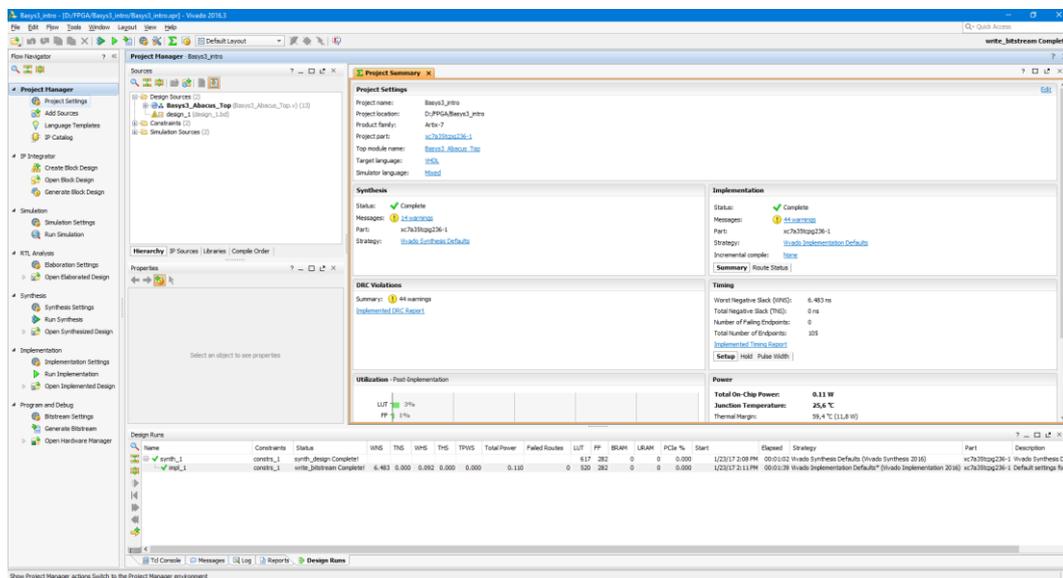
2.3 Programas empleados

2.3.1 Vivado

Para realizar la implementación del algoritmo se empleará el software diseñado por Xilinx para diseñar los circuitos digitales a programar en sus FPGAs[15].

Concretamente se empleará la distribución *Vivado HL WebPACK* en su edición 2016.3, limitada a una serie de FPGAs concretas entre las que se incluyen las de la familia Artix 7.

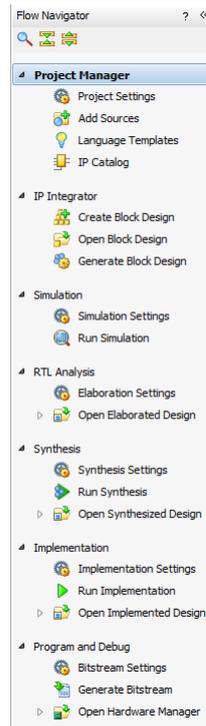
A continuación se muestra la interfaz gráfica del mismo:



25 - Interfaz gráfica de Vivado WebPack 2016.3

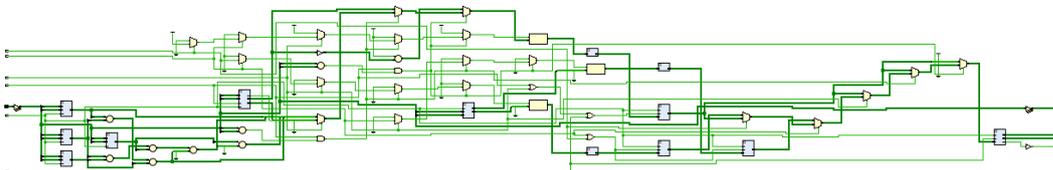
La interfaz tiene 3 secciones principales:

- Navegador de flujo de diseño: Permite controlar el contenido que muestra el programa.



26 - Navegador de flujo de diseño

- La primera opción “*Project Manager*” da acceso a la edición de los ficheros .vhd y .xdc en los que se indican la configuración hardware del diseño.
- La segunda opción “*IP Integrator*” permite realizar un diseño en base a bloques.
- El tercer apartado da acceso a la simulación de la lógica programada.
- El siguiente apartado “*RTL Analysis*” permite visualizar el diseño a nivel de registros:



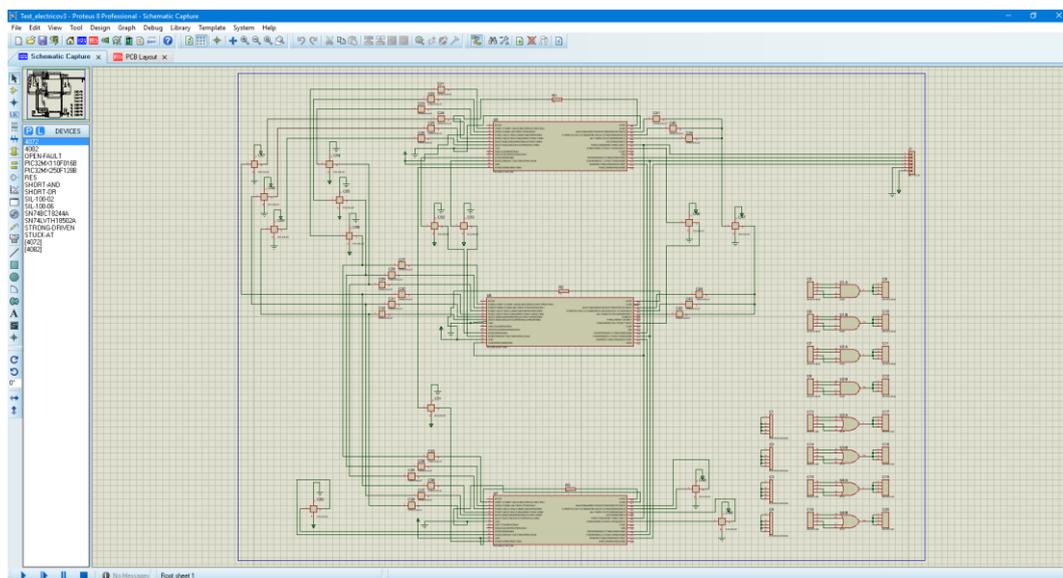
27 - Muestra de un diseño a nivel RTL

- “*Synthesis*” es el paso previo a realizar la implementación definitiva en la FPGA. Verifica el diseño VHDL y lo optimiza.
- “*Implementation*” adecua el diseño a la FPGA a programar

- El último apartado “Program and Dbug” da acceso a la generación de la cadena de bits empleada para programar la FPGA, así como a diversas herramientas de depuración.
- La consola aparece en la parte inferior de la pantalla, da información sobre las diferentes acciones ejecutadas, así como sobre los fallos y errores presentes en el diseño.
- La ventana principal, ocupa la mayor parte de la pantalla, es donde se programa y visualiza la mayor parte de la información.

2.3.2 Proteus 8

Para realizar el diseño del circuito generador de defectos eléctricos se emplea la versión 8 de Proteus.



28 - Interfaz gráfica en Proteus 8

En este caso, será necesario realizar conexiones entre los circuitos, así como emplazarlos en una placa base.

Para el diseño de los circuitos y las conexiones entre los mismos se emplea el modo ISIS, mientras que para emplazar los componentes se emplea el modo ARES. Existe la opción de ver el diseño en 3D con el modo 3D Visualizer.



29 - Modo ISIS, ARES y 3D Visualizer en Proteus 8

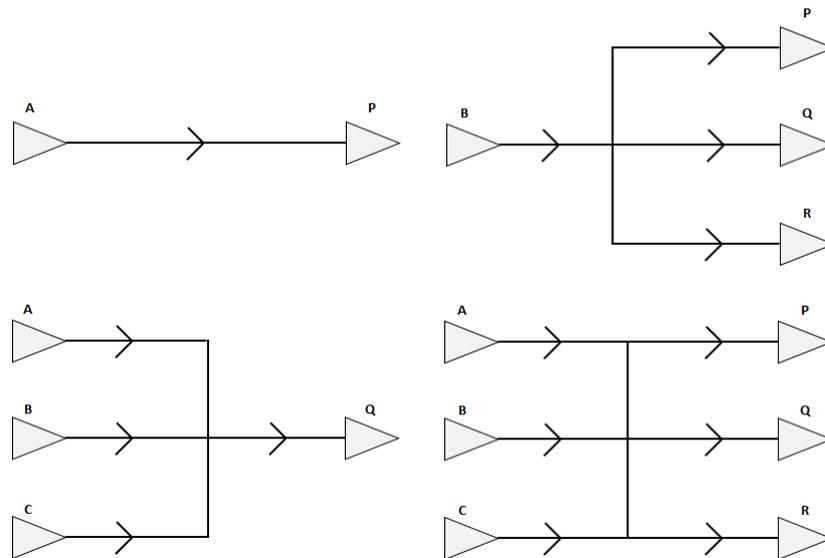
En el caso de los modos ISIS y ARES, se nos presentan a la izquierda en una lista los componentes disponibles, mientras que en la zona principal tratamos con el circuito en cualquiera de sus versiones, *schematic* o *layout*.

El modo 3D Visualizer simplemente permite comprobar de manera más realista la colocación de los componentes, así como sus dimensiones.

3 Placa de pruebas

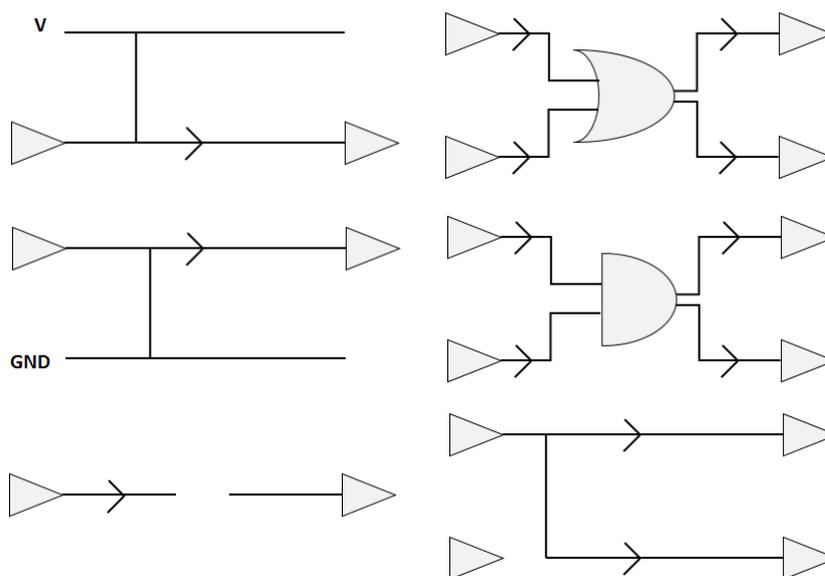
3.1 Concepto

Como ya se ha expuestos, existen cuatro grandes tipos de pistas, redes sencillas, redes con múltiples buffers de salida, redes con múltiples buffers de entrada y redes con varios buffers tanto de salida como de entrada.



30 - Posibles tipos de redes

También conocemos todos los posibles defectos eléctricos deterministas que pueden ocurrir de manera aislada sobre una pista:



31 - Defectos eléctricos deterministas

Sobra recordar que en los casos donde una red esté controlada por más de una entrada, habrá siempre buffers triestado, que también pueden estar presentes en redes sencillas.

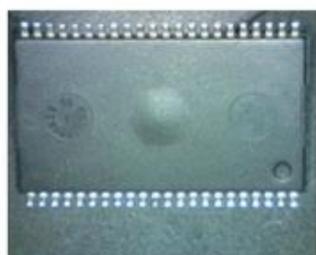
La placa diseñada ha de contar con modelos de los cuatro grupos de redes. También ha de permitir generar todos los defectos planteados en cada una de ellas, por lo que la placa ha de ser completamente configurable.

La placa diseñada ha de permitir una conexión sencilla a la plataforma de desarrollo, la Basys 3, por lo que su interfaz ha de ser un conector macho que encaje en el puerto PMOD de la Basys 3[16].

3.2 Protección ante cortocircuitos

El riesgo que presenta permitir una libre configuración de la placa es que se pueden causar cortocircuitos que dañen los integrados, la placa, o incluso la FPGA y plataforma de desarrollo.

Los riesgos que implica un cortocircuito pueden ser variados, pero están causados por el estrés térmico al que se somete una zona concreta de un circuito[17]. Estos efectos se pueden llegar a apreciar en el exterior de los circuitos integrados en forma de bultos, agujeros, quemaduras o grietas.



Package bulge



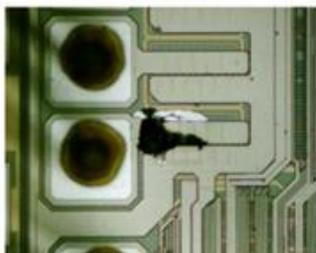
Package hole



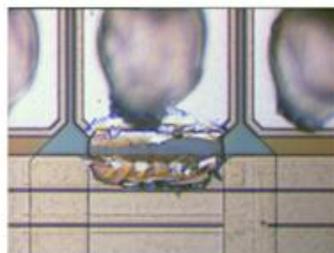
Package burnt/cracked

32 - Daños visibles provocados por cortocircuitos en circuitos integrados. Fuente [17]

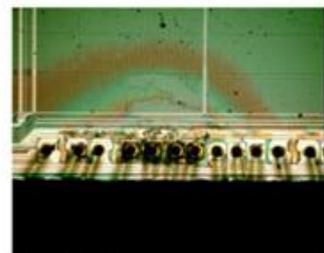
Sin embargo, hay ocasiones en las que el daño causado por un cortocircuito no es apreciable a simple vista, causando diversos efectos, como se puede apreciar en la siguiente imagen:



Burnt metal



Open connection



Heat stress



Melted bond wire

33 - Daños internos de los circuitos integrados provocados por cortocircuitos. Fuente [17]

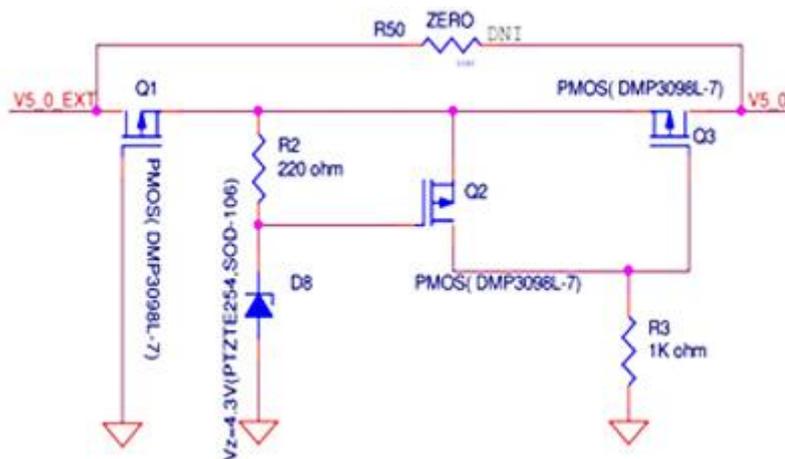
Estos fallos se deben a la gran cantidad de corriente fluyendo a través de un punto concreto, causante de una elevada temperatura capaz de ocasionar los defectos mostrados previamente. Estos fallos pueden aparecer tanto por picos de corriente elevados en lapsos de tiempo muy cortos, como en niveles de corriente ligeramente elevados pero mantenidos en el tiempo.

Algunas causas de levados picos de corriente son un consumo excesivo de corriente, una baja resistencia entre la alimentación y tierra, un cortocircuito entre tensión y tierra, o un fallo en un circuito integrado.

Es por ello que se ha de encontrar un modo de poder provocar los cortocircuitos sin dañar los componentes ni la placa, pues los defectos eléctricos que se quieren tratar incluyen los casos expuestos, como una baja resistencia entre alimentación y tierra.

Las opciones disponibles son cuatro:

- **Diodos:** Un diodo en serie a la línea que se quiere proteger, para proteger las pistas de corriente inversa. Esta solución causa una pérdida aproximada de 0.5 V en la línea por cada diodo presente en la misma, y no llega a ofrecer protección completa ante cortocircuitos.
- **Puente rectificador:** Se debería de colocar en serie a la línea a proteger. Las pérdidas de tensión en este caso son el doble, pues la corriente pasaría a través de dos diodos. Pese a que tampoco llega a ofrecer protección completa ante cortocircuitos, se acerca más al comportamiento que buscamos, pues permite la circulación de corriente en ambos sentidos.
- **Circuito MOSFET:** Para evitar la elevada caída de tensión de los circuitos previos se puede emplear un diseño en base a transistores MOSFET. Sin embargo, esta solución sigue sin ofrecer una protección efectiva ante cortocircuitos.



34 - Circuito MOSFET de protección propuesto por Banakar y Mathew en [17]

- **PolyZen:** Una combinación de diodo zéner y un fusible reconfigurable en paralelo, lo que permite limitar la corriente que circula mientras se mantiene la tensión. Cuando la corriente es muy elevada el fusible interrumpe el paso de corriente, por lo que resultaría efectivo ante cortocircuitos. Sin embargo estos dispositivos reducen la tensión de la línea[18].

Dispositivo de protección	Protección ante cortocircuitos	Caída de tensión en la línea ($I = 200 \text{ mA}$)[17]	Coste orientativo
Diodo	NO	0.5 V	0.10 €
Puente rectificador	NO	1 V	0.18 €
Circuito MOSFET	NO	50 mV	0.80 €
PolyZen	SI	200 mV	1.20 €

35 - Resumen de opciones para proteger un dispositivo del estrés térmico

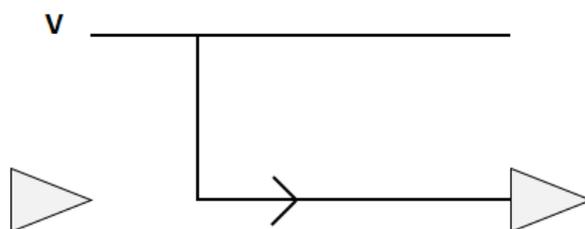
Como podemos ver, la única opción disponible para proteger la placa de cortocircuitos, son los polyzen. Sin embargo, el influir sobre el funcionamiento normal del circuito causando caídas de tensión, unido al coste que tendría implementar esta protección en todas las líneas de la placa, hace que esta protección no sea viable.

Como se busca diseñar una placa que permita ser empleada varias veces y con múltiples configuraciones, se ha de buscar un método de protección ante cortocircuitos que no modifique el funcionamiento del mismo.

La única opción disponible es la simulación de los defectos en la propia placa:

- Stuck-at one:

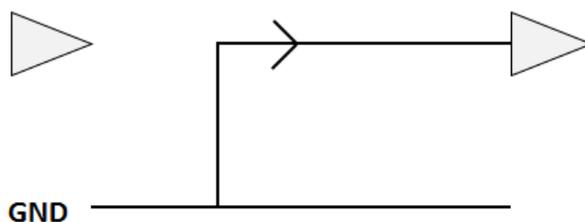
Este fallo se simulará separando las celdas de entrada a la red de las celdas de salida, y llevando estas últimas al valor lógico 1:



36 - Simulación del fallo stuck-at one

- Stuck-at zero:

Este fallo, al igual que el anterior requiere aislar las celdas de entrada a la red para evitar causar cortocircuitos. Para simular el comportamiento se conectará la celda de salida a un valor de tensión al valor lógico 0:



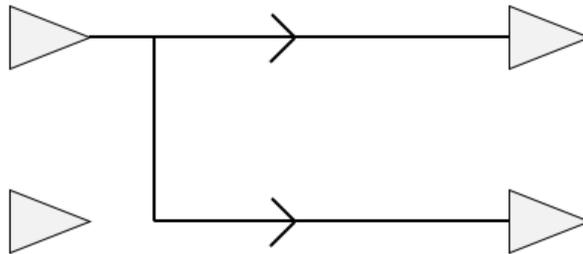
37 - Simulación del fallo stuck-at zero

- Stuck open: Este tipo de fallo no requiere simulación. Su comportamiento se recrea aislando las celdas de entrada de las de salida.



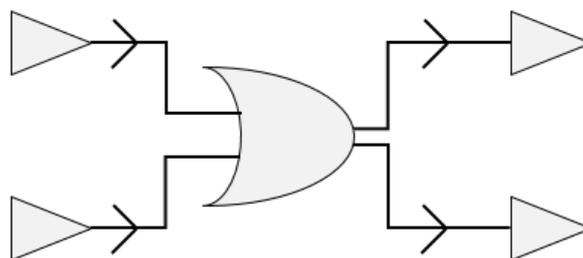
38 - Generación del fallo stuck open

- Strong-driven short: El comportamiento de este cortocircuito se simula aislando las celdas de entrada de tipo *weak-driven*, es decir, las que no influyen sobre el estado de la red:



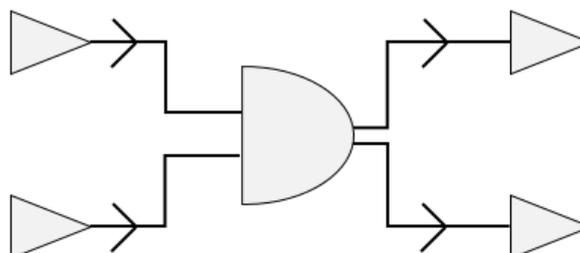
39 - Simulación del fallo strong-driven short entre dos redes

- OR-type short: Simular este fallo requiere aislar todas las celdas de la red además del uso de un componente adicional. Se simula con una puerta lógica de tipo OR, a cuyas entradas se conectarán las celdas lógicas de entrada que formen parte del circuito, y cuya salida serán todas las celdas de salida:



40 - Simulación del fallo OR-type short entre dos redes

- AND-type short: La simulación de este fallo es análoga al OR-type short, pero empleando una puerta lógica tipo AND:



41 - Simulación del fallo AND-type short entre dos redes

3.3 Circuitos considerados

La cadena de *boundary-scan* está formada por las diversas celdas que contengan los circuitos integrados a probar. Para que el test eléctrico funcione dichos dispositivos han de tener habilitado el test JTAG además de ser completamente compatibles con el estándar IEEE 1149.1[1].

Los circuitos que se ha valorado incluir en la placa son los siguientes:

- Microcontroladores ATMEL de 8 bits: Dispositivos como el ATMEL ATMEGA 324A-PU[19] o el ATMEGA162-16PU[20]. Con coste aproximado de 6€ la unidad[21], [22]. Permiten conexión a la placa por medio de zócalo. Presentan el inconveniente de permitir cambiar el estado del fusible habilitador del test JTAG mientras se realiza el propio test.
- Transceptor de bus universal: El SN74LVTH18502APM[23] o SN74LVTH18504APM[24] de Texas Instruments. Con un precio orientativo de 10€ la unidad[25]. Presentan el inconveniente de ser dispositivos de soldadura superficial, la poca distancia entre pines dificulta su correcto montaje. No obstante, su función permite una comprensión más sencilla de su funcionamiento, así como de las celdas del registro *boundary-scan*.
- Transceptor de bus universal: Al igual que el transceptor de bus, es de montaje superficial. Los modelos valorados son el SN74BCT8244ADW[26] y el SN74BCT8240A [27] con un precio unitario aproximado de 7.5€[28].
- Microcontroladores MICROCHIP de 32 bits: Se contempla la familia PIC32MX, concretamente el PIC32MX110F016B-I/SP[29] y el PIC32MX250F128B-50I/SP[29]. Estos componentes tienen un precio medio de 3€ la unidad[30], [31], y permiten su montaje sobre zócalo.

Entre estas opciones, teniendo en cuenta la facilidad de soldadura y recambio, el coste económico, y la complejidad del fichero BSDL correspondiente, se ha decidido emplear los dos modelos de microchip.

3.4 Listado de componentes

Para realizar la conexión entre la placa y la plataforma de desarrollo se emplearán conectores de 6 contactos en ángulo recto.

Para poder configurar los fallos de tipo stuck-at se emplearán conectores placa a placa rectos, sin ángulo, junto a unos jumpers de dos contactos que marcarán el comportamiento de esa red.

Para poder configurar fallos de tipo cortocircuito son necesarios unos cables puente hembra, para poder conectar las redes a las puertas lógicas.

Las puertas lógicas empleadas serán de dos tipos, OR y AND, en ambos casos admiten un máximo de hasta cuatro entradas.

En cuanto a los circuitos empleados en el test, se emplean dos unidades del PIC32MX110F016B-I/SP y una unidad del PIC32MX250F128B-50I/SP, todos ellos irán colocados sobre sus zócalos correspondientes.

El listado de componentes empleados se indica a continuación, el código indicado se corresponde a la tienda Farnell - Element 14[32].

Denominación	Tienda	Código	Uds.	Precio	Coste
TMM-106-01-T-S-RA	Farnell	2308468	2	0.698	1.396
WURTH ELEKTRONIK 61304011121	Farnell	2356175	5	1.2915	6.4575
3M 969102-0000-DA	Farnell	2579814	40	0.0568	2.272
PSG-JMP150FF	Farnell	2452748	4	2.39	9.56
PIC32MX110F016B-I/SP	Farnell	2467671	2	1.92	3.84
PIC32MX250F128B-50ISP	Farnell	2313768	1	4.18	4.18
DIP 28 1-2199299-2	Farnell	2453476	3	0.376	1.128
DIP 14 1-2199298-3	Farnell	2445621	4	0.123	0.492
CD4082B	Farnell	1470852	2	0.414	0.828
CD4072B	Farnell	1470848	2	0.352	0.704
				TOTAL	30.85€

42 - Lista de materiales

Como podemos ver, el coste final de la placa es de 30.85 €.

3.5 Diseño

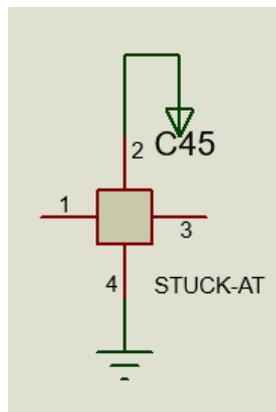
El diseño de la placa se realizó en Proteus 8. El mismo se encuentra disponible anexo en formato digital.

Para realizar el diseño de la placa fue necesario crear los dos circuitos integrados a emplear en la cadena de *boundary-scan*, pues la librería no contaba con sus modelos.



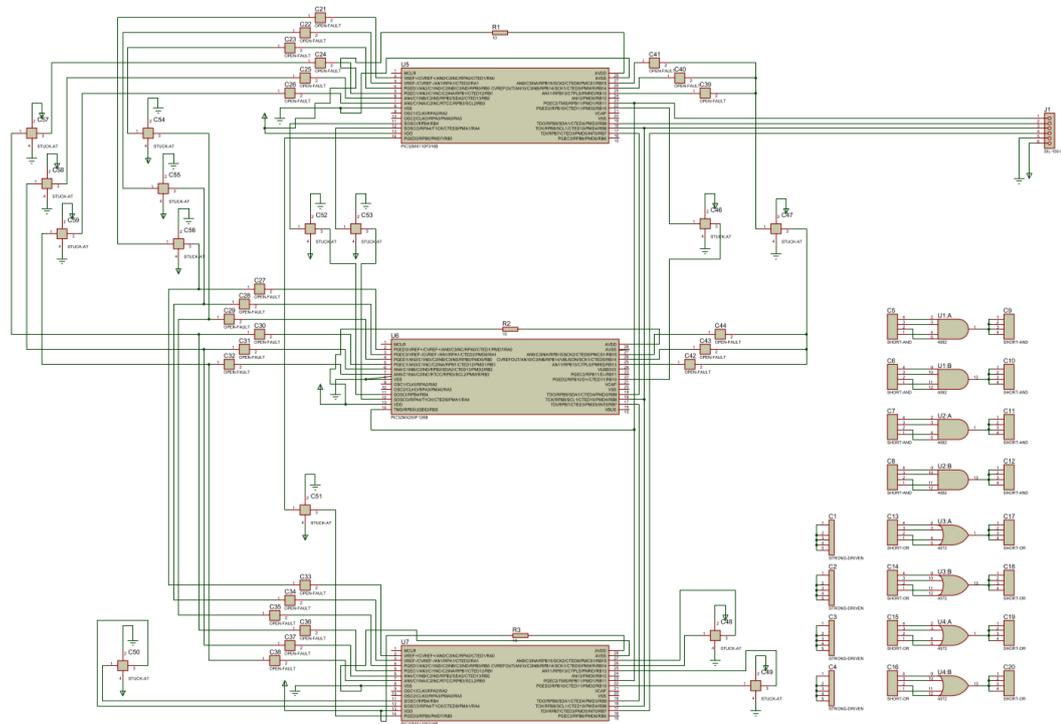
43 - Modelos de PIC32MX en Proteus 8

Para facilitar el diseño de la placa también se crearon los bloques correspondientes a la configuración de los defectos tipo stuck-at.



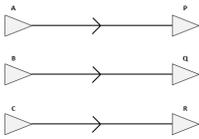
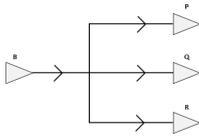
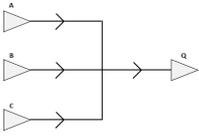
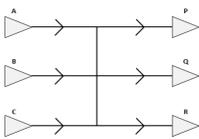
44 - Modelo de bloque empleado para provocar el defecto stuck-at

Como se puede ver, los bloques para generar los fallos de tipo short, se encuentran aislados del diseño principal, esto permite mayor libertad en el momento de generar los defectos deseados.



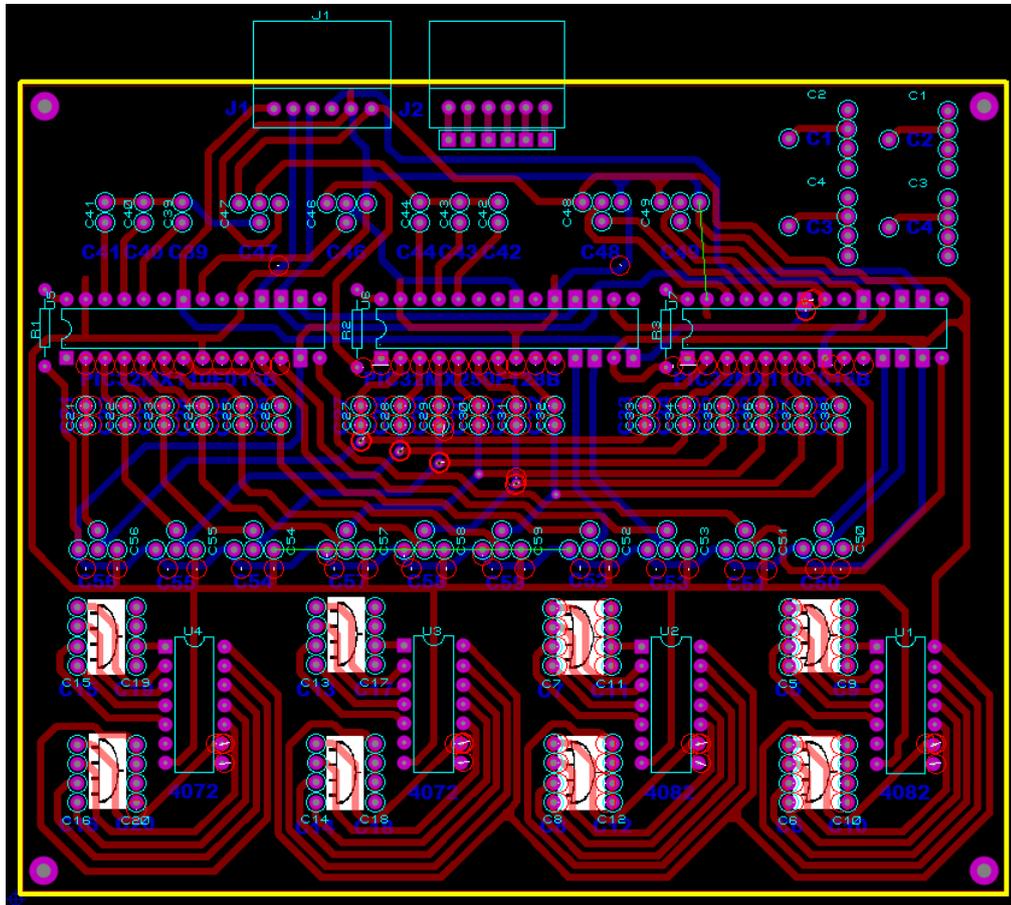
45 - Diseño de la placa en ARES

Teniendo en cuenta la variedad de pistas posibles presentadas anteriormente, las pistas presentes en el diseño son:

Tipo de red	Integrado en origen	Número de patilla	Conector Open	Integrado en destino	Número de patilla	Conector open	Conector stuck-at y shorts
	5	11	NA	6	11	NA	C52
	5	12	NA	6	12	NA	C53
	5	14	NA	7	14	NA	C51
	5	21	NA	6	21	NA	C46
	7	11	NA	7	12	NA	C50
	7	21	NA	7	24	NA	C49
	7	25	NA	7	26	NA	C48
	5	2	C21	6	2	C27	C56
				7	2	C33	
	5	3	C22	6	3	C28	C55
				7	3	C34	
	5	4	C23	6	4	C29	C54
				7	4	C35	
	6	5	C30	5	5	C24	C57
	7	5	C36				
	6	6	C31	5	6	C25	C58
	7	6	C37				
	6	7	C32	5	7	C26	C59
	7	7	C38				
	5	24	C39	6	24	C42	
	5	25	C40	6	25	C43	C47
	5	26	C41	6	26	C44	

12 - Configuraciones de pistas presentes en la placa

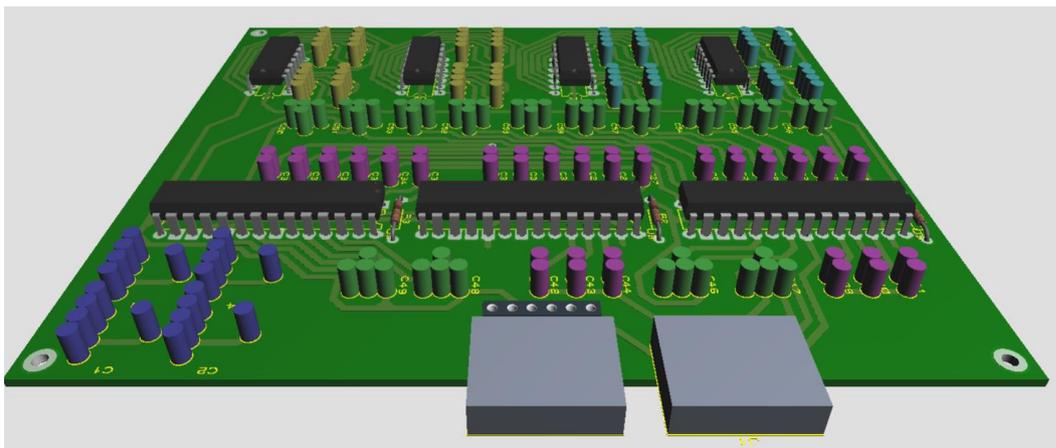
Una vez se tiene listo el diseño en ARES, se realiza el diseño del layout sobre la placa, para lo que se emplea ISIS:



46 - Placa con los componentes emplazados. Vista del layout

El layout necesario para realizar el revelado de la placa se encuentra anexo, se trata de “Placa defectos eléctricos - Layout.pdf”.

Si nos fijamos en el modelo 3D de la placa podemos observar los diferentes elementos:

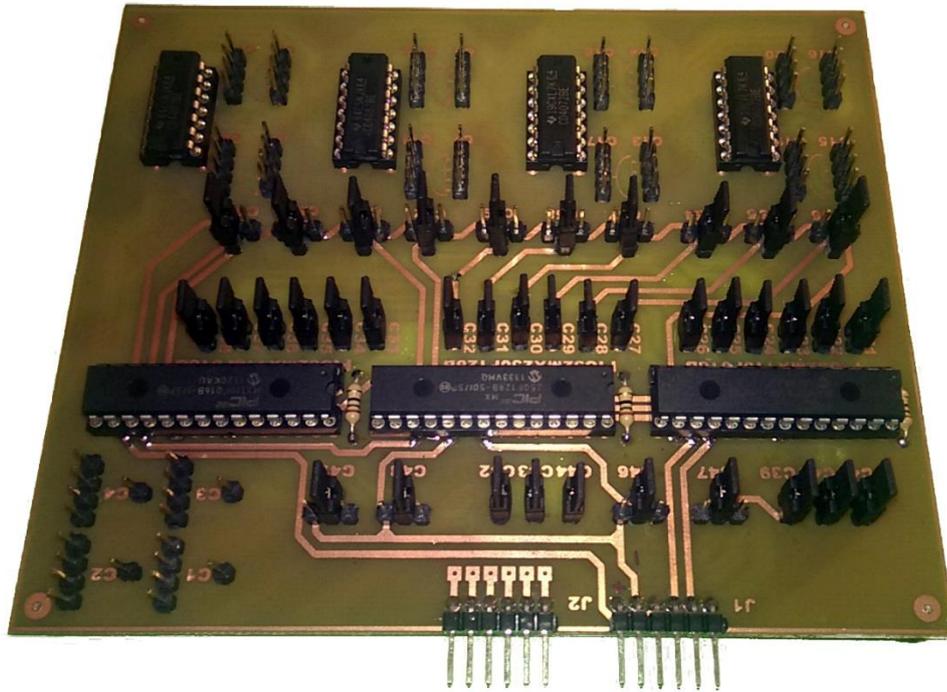


47 - Visualización 3D de la placa

- Conectores de la placa a la Basys 3, se trata de los dos conectores grises que sobresalen en el primer plano.
- Microcontroladore PIC32MX, son los 3 circuitos integrados alineados, el PIC32MX250 es el central.
- Puertas lógicas AND y OR, se trata de los 4 circuitos integrados situados en paralelo.
- Generadores de fallo *stuck-open* en buses de línea, son los conectores de color morado.
- Generadores de fallo tipo *stuck-at* y *stuck open*, se trata de los conectores verdes.
- Los conectores azules indican el subcircuito encargado de recrear los fallos *strong-driven shorts*.
- Los conectores amarillos se corresponden a la simulación de los fallos *OR-type shorts*.
- Los conectores cian se corresponden a la simulación de los fallos *AND-type shorts*.

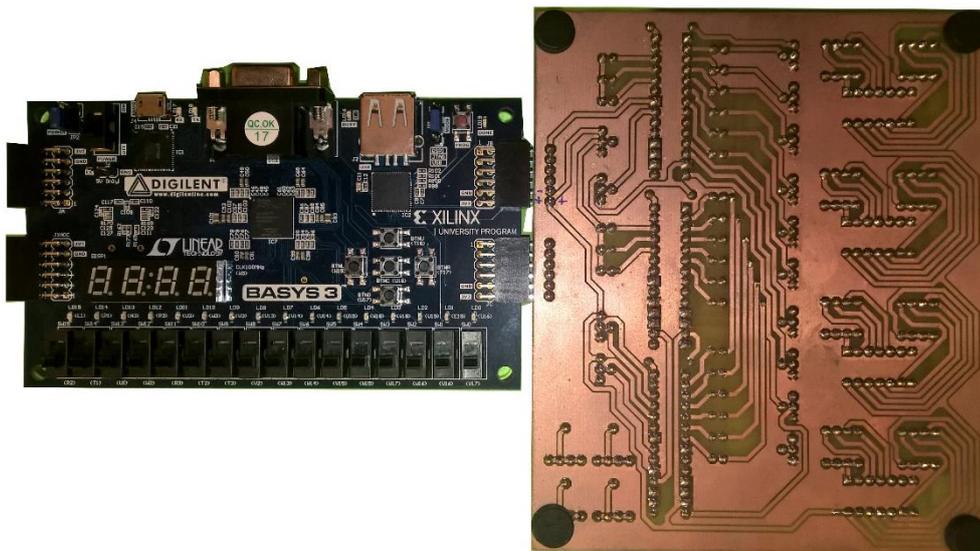
3.6 Ejecución y configuración

Una vez realizado el proceso de revelado y soldadura, el resultado es el siguiente:



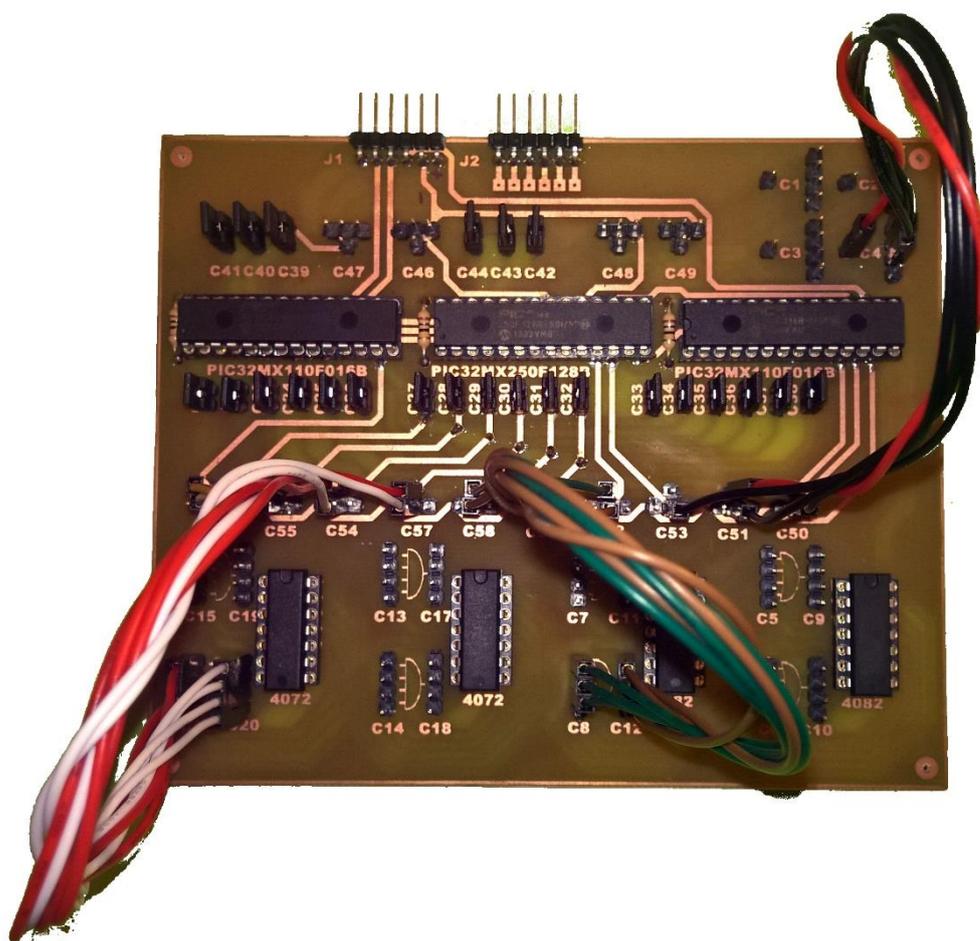
48 - Placa de test finalizada

La unión a la Basys 3 se realiza de la siguiente manera:



49 - Unión entre la placa y la Basys 3

A continuación se muestra cómo generar los defectos de tipo short. Los cables rojos a la izquierda envían la señal de los conectores C54 a C57 a una puerta lógica AND que simula un fallo *AND-type short*, el resultado es enviado a los mismos conectores a través de los cables blancos. Los cables verdes envían la señal de los conectores C52, C58 y C59 para la generación del fallo *OR-type short*, cuyo resultado es enviado a través de los cables marrones a las celdas de salida de la red. Finalmente, el cable rojo a la derecha represente la celda *strong-driven* que gobierna un corto entre las redes correspondientes a los conectores C50, C51 y C53, la respuesta es enviada a esas redes por medio de los cables negros.

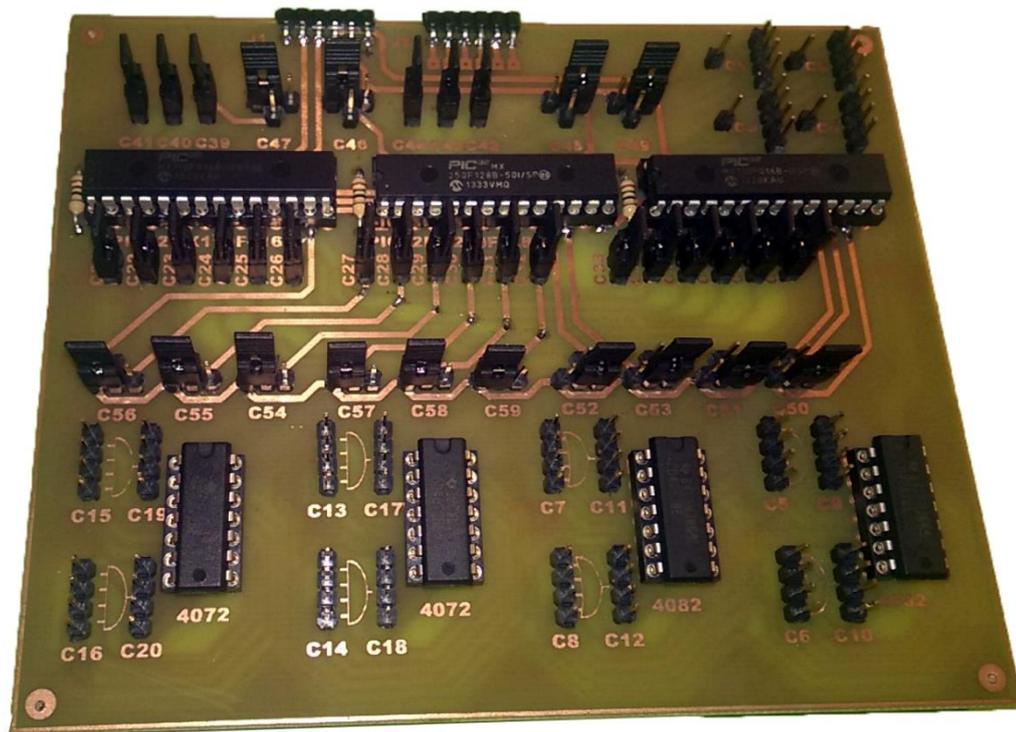


50 - Ejemplo de generación de fallos tipo short

Para generar los fallos tipo *stuck-open* en el caso de las redes completas, basta con retirar los jumpers de los conectores C46 a C59. En los casos de las redes con varias celdas de entrada y/o salida, también se pueden simular fallos

stuck-open que afecten en exclusiva a esa subnet, para ello se ha de retirar el conector correspondiente en C21 a C44.

Finalmente, la simulación de los fallos *stuck-at* se emplean los jumpers, en el siguiente ejemplo se aprecia como los conectores C54 a C59 están simulando un fallo *stuck-at one*, mientras que C50 a C52 están generando un fallo *stuck-at zero*.



51 - Muestra de generación de fallos tipo *stuck-at*

Sobra decir que se puede aplicar cualquier configuración imaginable de las aquí expuestas sobre la placa, tanto individualmente como al mismo tiempo.

4 Implementación

4.1 El test a realizar

Toda implementación ha de tener un objetivo, en este caso será la identificación de los defectos *stuck-at* presentes en la tarjeta diseñada en el capítulo 3.

El circuito a comprobar es compatible en su totalidad con el estándar IEEE 1149.1[1], lo que permite emplear la arquitectura *boundary-scan* para realizar el test de defectos eléctricos. Antes de ejecutar el test se hace necesario verificar el estado de la arquitectura *boundary-scan*, es decir, comprobar previamente el puerto de acceso para test (TAP), así como el registro *boundary-scan*.

4.1.1 Verificar la integridad de la arquitectura *boundary scan*

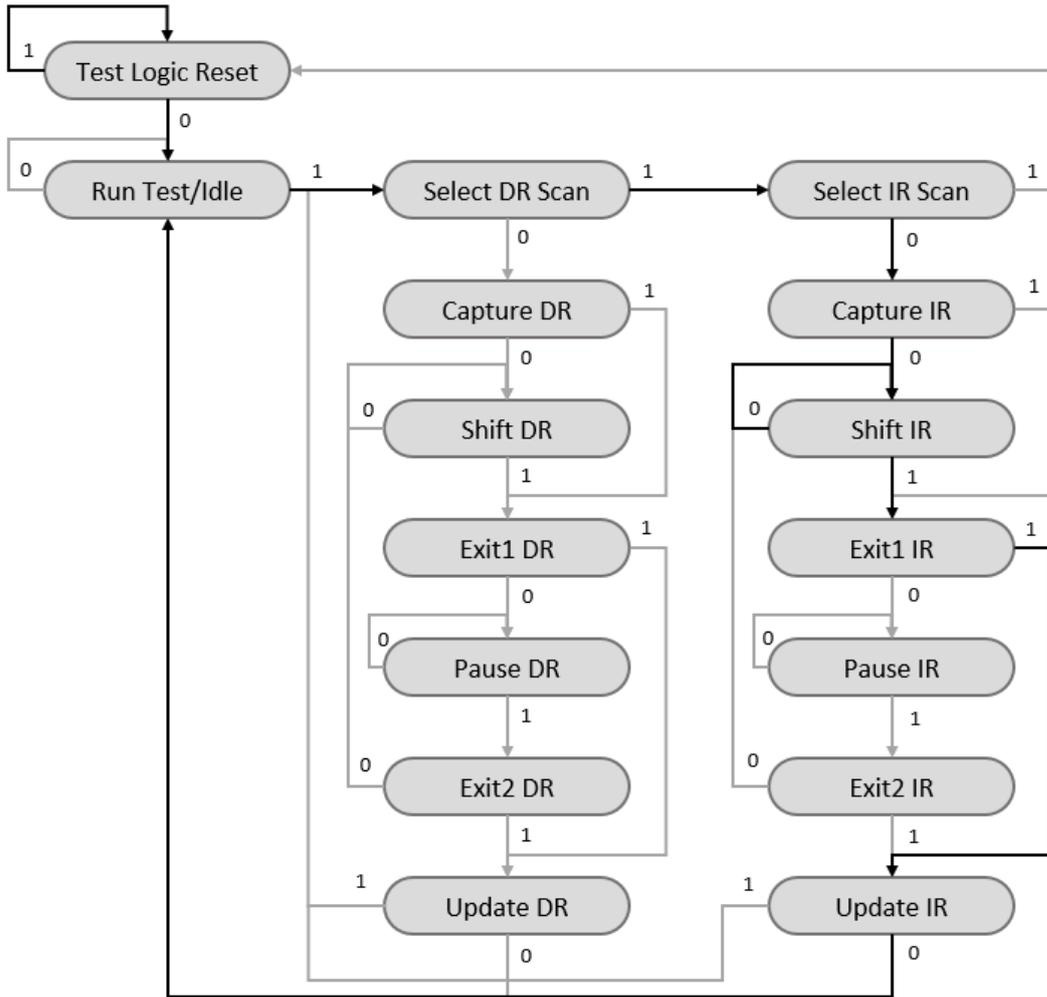
Se trata de una necesidad previa al test. Garantiza que el test se realiza correctamente, las celdas origen y destino seleccionadas son las correctas en el registro *boundary-scan*.

Para ello primero se ha de comprobar el correcto funcionamiento del puerto de acceso para test.

4.1.1.1 Puerto de acceso para test – TAP

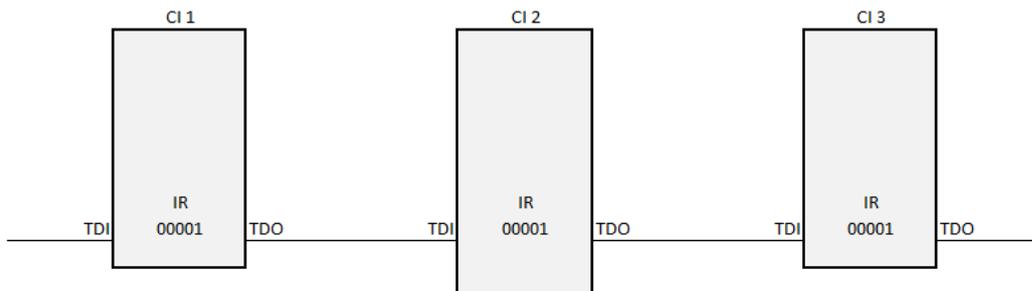
Una necesidad previa a ejecutar el test de defectos eléctricos. Asegura una correcta conexión del TAP entre todos los componentes que forman parte de la cadena *boundary-scan*. También asegura el correcto funcionamiento de las máquinas de estados finitas internas a cada uno de los integrados.

Para asegurar un correcto manejo y sincronización entre todas las máquinas de estados de los *TAP controllers*, se realiza un reset a la misma, para ello se envían 5 pulsos de reloj durante los que se mantiene TMS a 1. Esto coloca todos los controladores en el estado *Test Logic Reset*.



52 - Estados por los que pasa el TAP durante la verificación del puerto

Para realizar este test, se lleva el controlador TAP hasta la instrucción *Capture IR*. En este estado los registros de instrucción de los integrados cargar el código `INSTRUCTION_CAPTURE` indicado en sus ficheros BSDL. Los dos bits menos significativos de este código han de finalizar en 01 para ser compatibles con el estándar.



53 - Representación del estado de los registros de instrucciones tras *Capture IR*

A continuación, se pasa al estado *Shift IR*, por el que se pasa tantas veces como longitud total tenga el registro de instrucciones formado por todos los integrados presentes en la cadena. Cada vez que se pasa por este estado se realiza un desplazamiento de la cadena correspondiente a los registros de instrucciones.

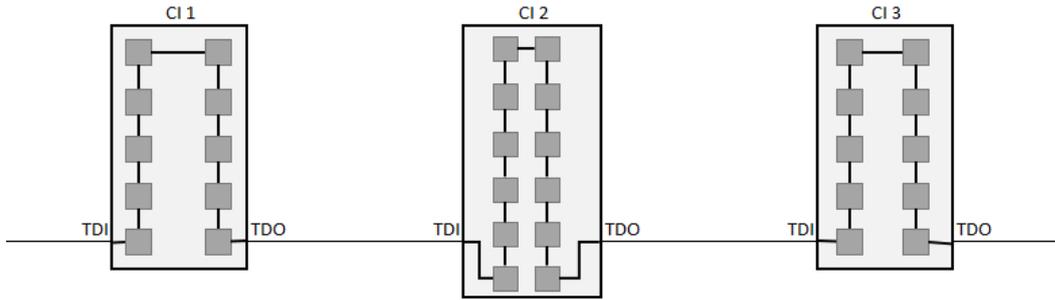
Se sigue llevando el controlador al estado *Exit1 IR*, lo que conlleva un último desplazamiento de la cadena. Se pausa el test llevando los controladores TAP al estado *Run Test/Idle*.

Si la cadena recibida en el registro interno de la FPGA coincide con el correspondiente a encadenar todos los códigos *INSTRUCTION_CAPTURE* en el orden correcto, se puede afirmar que no existe ningún problema en las conexiones correspondientes al TAP. También se comprueba que la captura e introducción de instrucciones funciona en todos los circuitos integrados. En el caso mostrado en la imagen 53, el valor del registro interno debiera corresponderse al código "000010000100001".

4.1.1.2 Registro *boundary-scan*

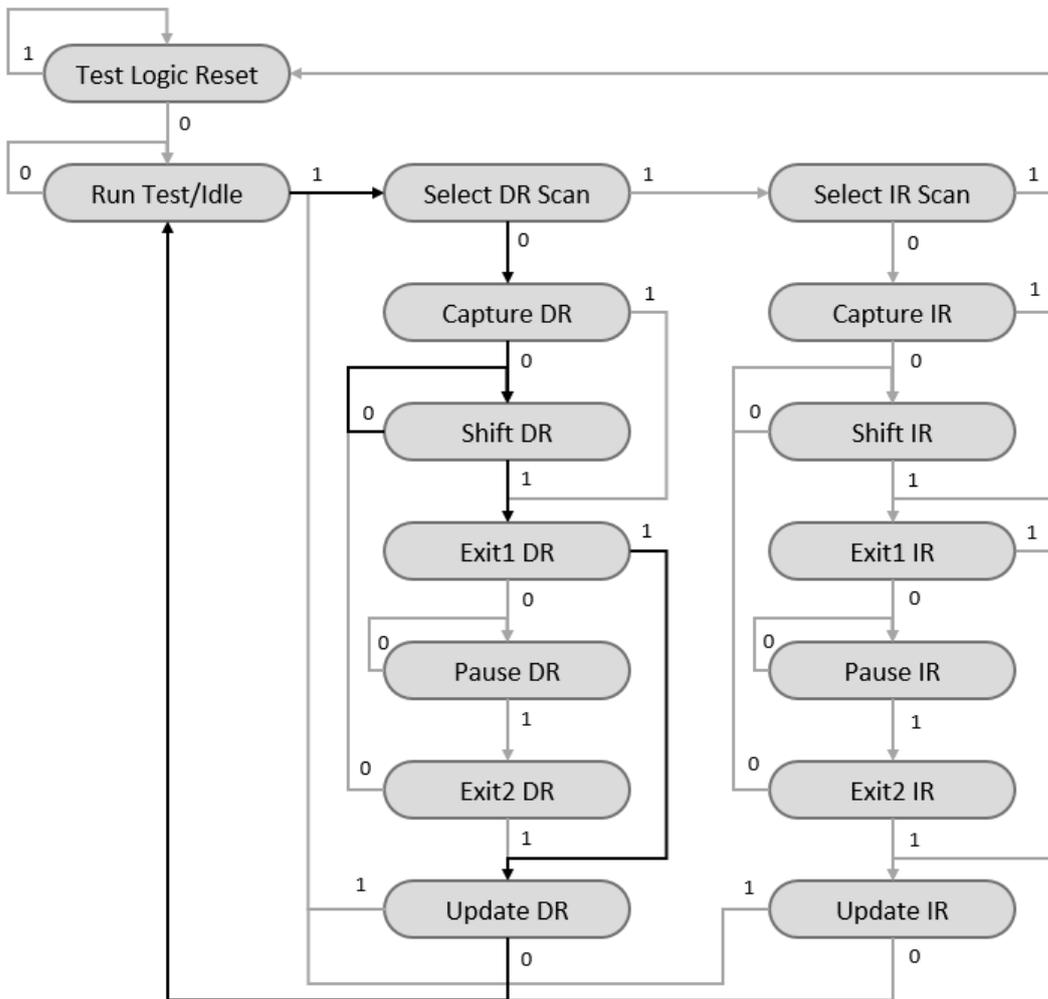
Ya comprobado el puerto de acceso para test, es preciso verificar la integridad de la cadena *boundary-scan*. Para ello se lleva la máquina de estados a *Capture IR* y se envía el código correspondiente a la instrucción *EXTEST* para todos los dispositivos por medio de los desplazamientos realizados durante *Shift IR* y *Exit IR*.

Se lleva el TAP a *Capture DR* y se precarga en el registro interno de la FPGA un código de igual longitud a la cadena *boundary-scan* compuesta por todos los circuitos. Se realiza un desplazamiento mediante *Shift DR* tantas veces como el doble de celdas que forman el registro *boundary-scan* menos uno. En el caso mostrado a continuación el total de desplazamientos realizados durante *Shift DR* ha de ser 63, pues el total de celdas que forman el registro es 32.



54 - Ejemplo de cadena boundary-scan

Se devuelve el controlador al estado *Run Test/Idle*, si los valores ocupados en el registro interno de la FPGA se corresponden con el patrón enviado al inicio, se puede asegurar un correcto funcionamiento del registro boundary-scan.

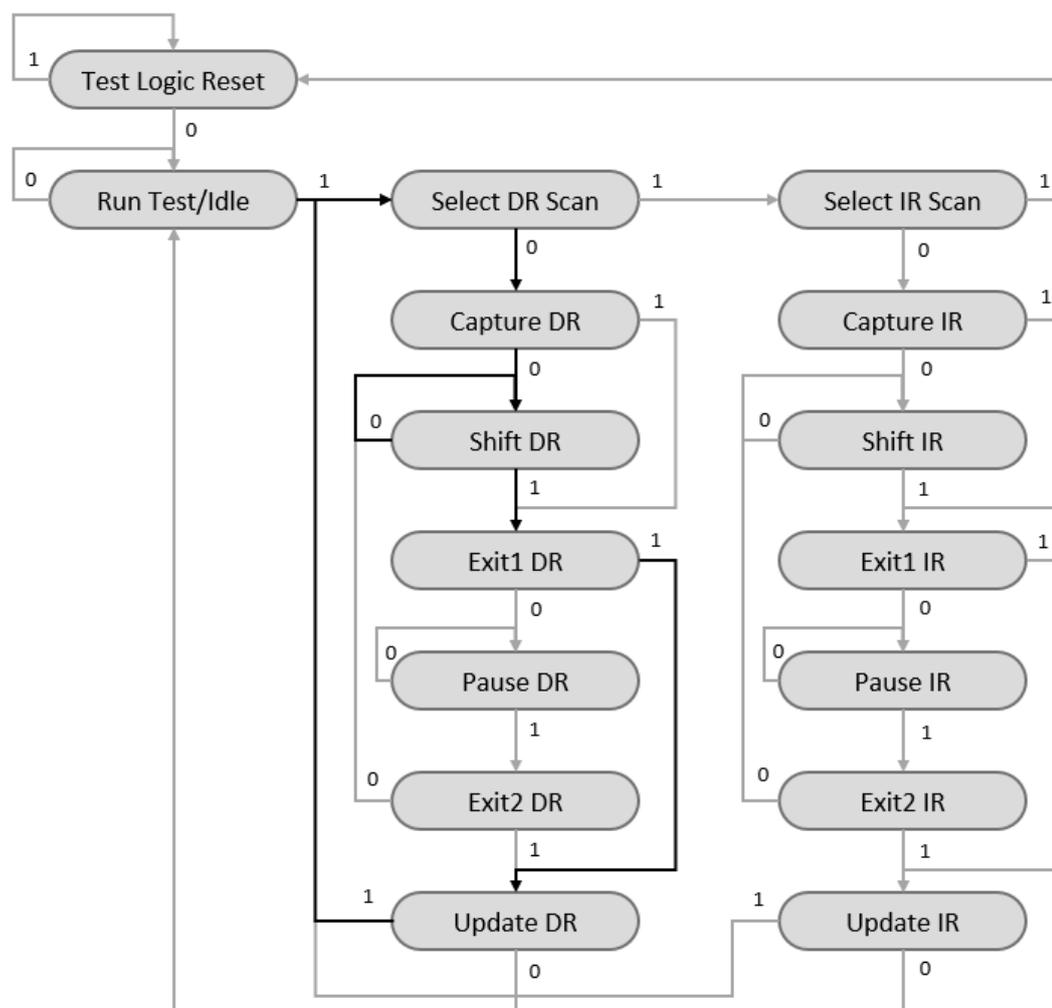


55 - Estados por los que pasa el TAP durante la verificación de la cadena boundary-scan

4.1.2 Verificar la interconexión entre pistas

El siguiente paso es la comprobación del estado en que se encuentran las interconexiones de la placa.

El funcionamiento será similar a la verificación del registro *boundary-scan*. Con una diferencia, puesto que ya está precargada la instrucción EXTEST de la verificación previa, no es necesario modificar el registro de instrucciones.



56 - Estados por los que pasa el TAP durante el test de defectos eléctricos

Se generará en la FPGA un vector de test a comprobar. Este vector de test será enviado al pasar por el estado *Shift DR* tantas veces como celdas forman el registro *boundary-scan* menos uno.

En el paso por *Update DR* el vector cargado en las celdas del registro *boundary-scan* actualizarán el estado de las pistas a las que se encuentran conectadas.

En el siguiente paso por *Capture DR* estas mismas celdas actualizarán sus valores con el resultado del test realizado. Siendo este desplazado por medio de las instrucciones *Shift DR* a la par que se va sustituyendo por el siguiente vector a comprobar.

4.2 Diseño del algoritmo a implementar

Según el momento del desarrollo en que se realice un test, conviene emplear un algoritmo u otro. Una vez la placa esté lista, conviene hacer un test rápido que identifique si existe algún tipo de problema en la misma. De este modo se reduce el tiempo que pasan en fase de test las tarjetas que se encuentran en buen estado. Tras realizar este test, las tarjetas que hayan dado algún tipo de fallo son sometidas a test más largos y que permiten identificar claramente dónde se encuentra el fallo.

En los casos en los que sea necesario identificar qué fallo ocurre y dónde tiene lugar el mismo, si el equipo empleado para realizar el test no es capaz de hacer un test adaptativo, es recomendable emplear el algoritmo *Walking One*[2].

Como ya se expuso en la sección 2.1.4.4 el test "*Walking One/Walking Zero Algorithm*" es la opción idónea en cuanto a identificación de los defectos eléctricos sin que haya opción de fallo.

En [3] A. Hassan, J. Rajski y V. K. Agarwal hacen una propuesta de implementación a nivel hardware para realizar el diagnóstico de las interconexiones. Para ello emplean el algoritmo "*Walking One/Walking Zero Algorithm*" y analizan el resultado de cada celda al finalizar el test, tras lo que obtienen:

Valor al finalizar Walking Ones	Valor al finalizar Walking Zeroes	Defecto presente (redes pares)	Defecto presente (redes impares)
1	1	Sin defectos.	Sin defectos.
1	0	-	Corto tipo OR.
0	1	-	Corto tipo AND.
0	0	Fallo stuck-at. Corto tipo OR. Corto tipo AND.	Fallo stuck-at.

13 - Diagnóstico propuesto por[3]

Como se puede observar, el diagnóstico propuesto sólo realizaría una identificación aceptable en el caso de circuitos un número de redes impar.

No obstante, en cuanto a la generación de vectores, el algoritmo propuesto es ideal, motivo por el que será el empleado para ello.

En cuanto a la detección, en nuestro caso el algoritmo será implementado a nivel de hardware, y puesto que no se ha encontrado ningún algoritmo de una etapa que permita realizar una correcta identificación de los fallos, se ha buscado otra opción.

El método empleado no será el tradicional, que supone todas las interconexiones están realizadas correctamente. En su lugar, el algoritmo desarrollado supondrá defectuosas todas las redes, considerando todos los fallos posibles sobre las mismas. En vez de realizar una búsqueda de las pistas defectuosas, se identificará cuando un error no está presente sobre la misma.

Cuando en la celda de salida de una red se reciba un valor lógico de tipo uno se puede descartar de forma certera el fallo tipo *stuck-at-zero* en su red. Para verificar que una pista no presenta un fallo de tipo *stuck-at-one*, se procede de manera similar en cuanto esa red recibe un valor lógico de tipo cero.

Pese a que el presente trabajo fin de grado pretende identificar los fallos de tipo *stuck-at-one* y *stuck-at-zero*, la propuesta realizada permite la detección de otro tipo de fallos como los de cortocircuito. Para descartar el cortocircuito entre dos pistas cualesquiera, solo se ha de esperar por un vector de test que cause diferente comportamiento en dichas pistas. Esto ocurrirá siempre que la interconexión no sea defectuosa debido a que el algoritmo genera vectores diagonalmente independientes.

Esta concepción del algoritmo permite considerar las subredes como una entidad propia, sin reducirlas a su red, lo que permite identificar fallos presentes en una sola de las ramas.

4.3 Desarrollo del algoritmo

Para el desarrollo del algoritmo se emplean diferentes registros:

Registro	Utilidad
TOTAL_BSC	Total de celdas Boundary scan.
TOTAL_NETS	Total de redes.
IR_LENGTH	Longitud total de la cadena IR.
JTAG_STATUS	Estado de la cadena JTAG.
BS_STATUS	Estado de la cadena BS.
IR_CAPT_OK	Código correcto de IR al ejecutar CAPTURE IR tras Test Logic/Reset.
TOTAL_O_CELLS	Total de output cells (entradas a ICs).
TOTAL_I_CELLS	Total de input cells (salidas de ICs).
MAX_PAR_IN	Indica el máximo de input cells en paralelo.
TOTAL_C_CELLS	Total de control cells (internas a los ICs).
O_CELLS_NETS	Colum1: Red. Colum2: BSC tipo output. Ordenado por redes.
I_CELLS_NETS	Colum1: Red. Colum2: BSC tipo input. Ordenado por redes.
C_CELLS_BSC	Colum1: BSC controlada. Colum2: Enable BSC. Colum3: Valor enable.
W1W0	Indica si se está realizando el test W0 o W1.
STUCK_AT_1	Indica subredes con STUCK AT 1.
STUCK_AT_0	Indica subredes con STUCK AT 0.

14 - Registros estáticos empleados

Desplazador	Utilidad
IR_CHAIN	Cadena IR en FPGA
BSC_CHAIN	Cadena DR en FPGA
W1W0_NETS	Valores del test W1/W0 a realizar

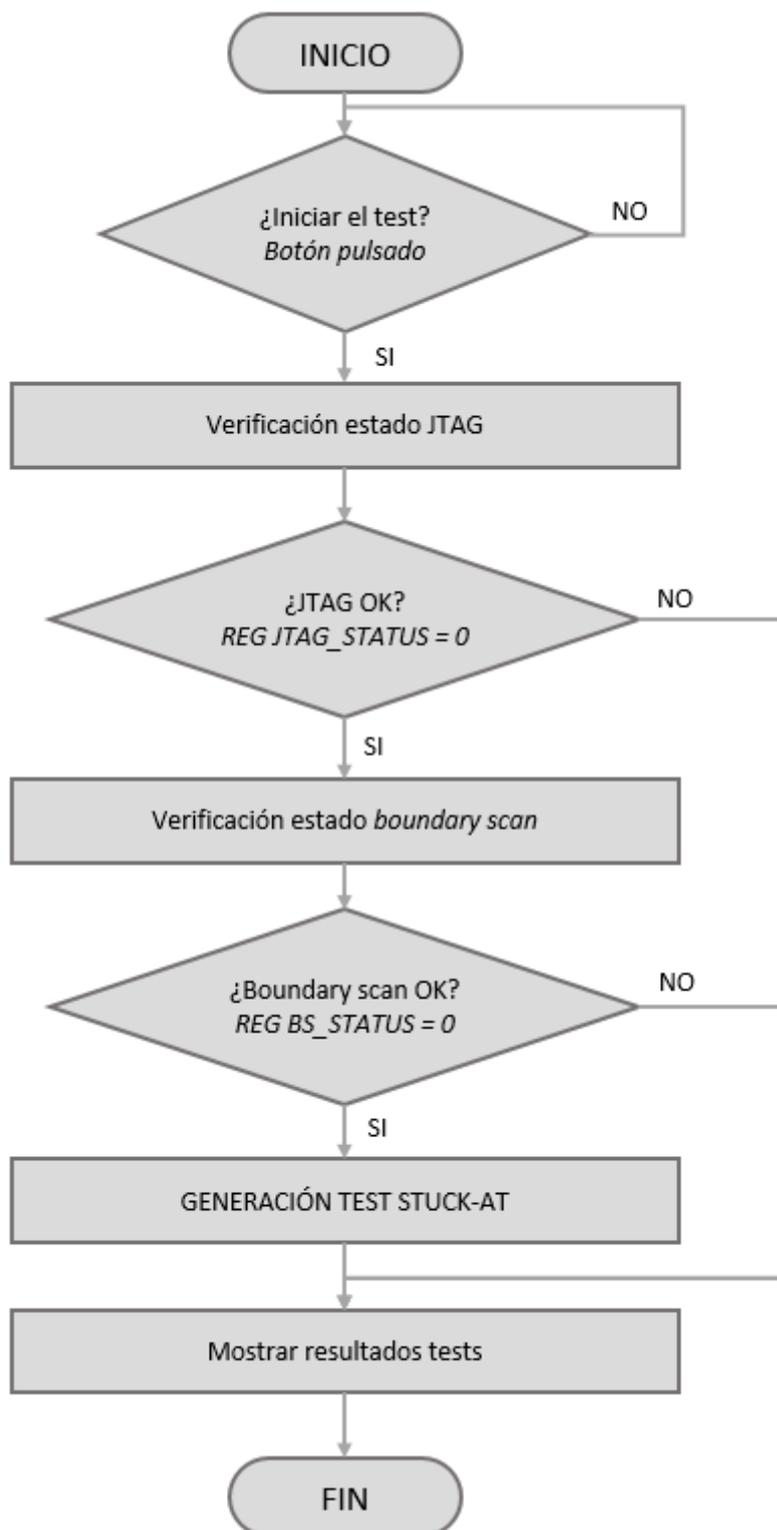
15 - Registros desplazadores empleados

También es necesario el uso de contadores:

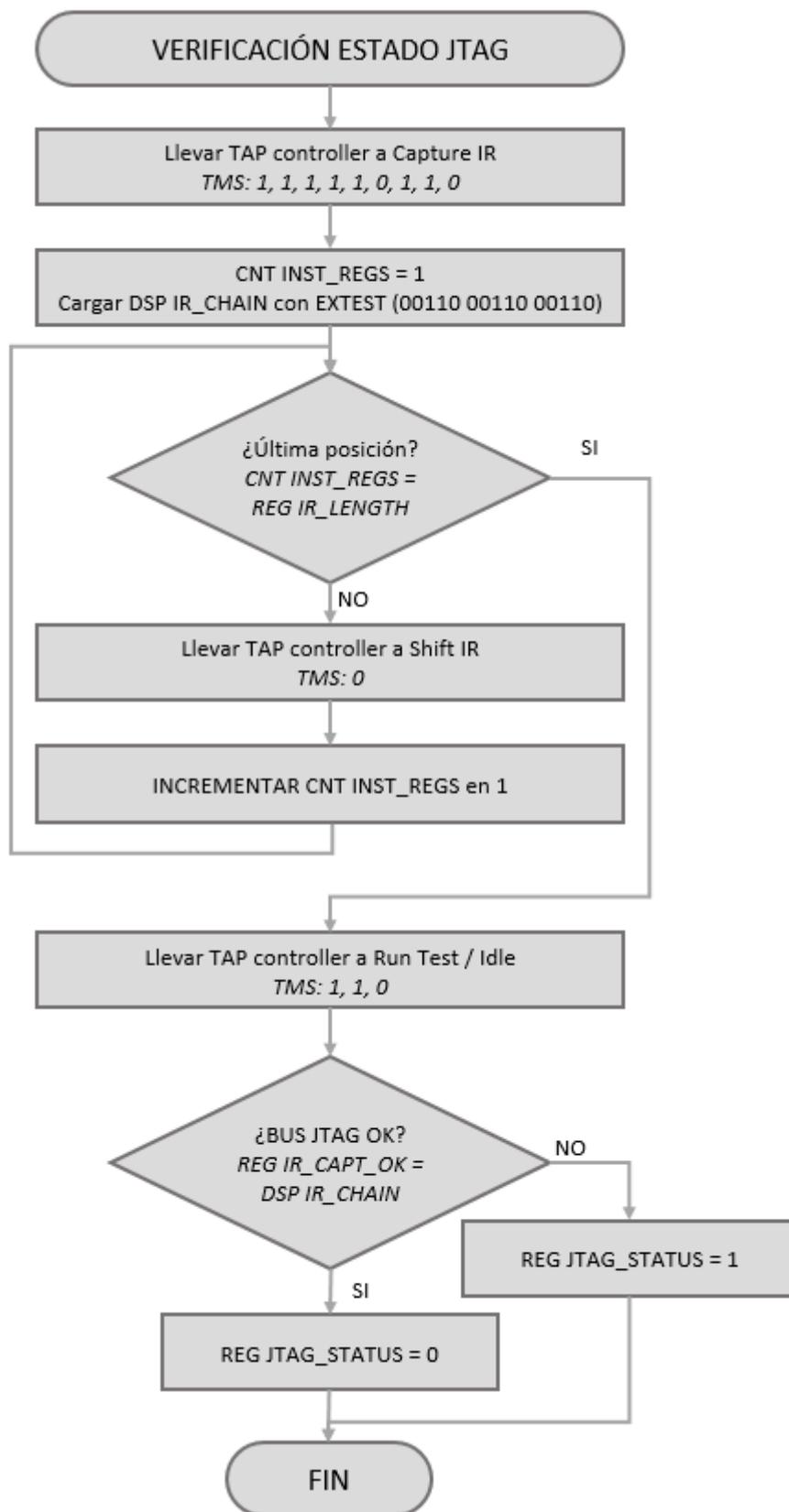
Contador	Utilidad
FAULTS	Recorrer REG OPEN_FAULTS, REG STUCK_AT_1 y REG STUCK_AT_0
INST_REGS	Recorrer SHIFT IR y EXIT IR correctamente
DATA_REGS	Recorrer SHIFT DR y EXIT DR correctamente
OUT_CELLS	Recorrer output cells
IN_CELLS	Recorrer input cells
CTRL_CELLS	Recorrer control cells
PAR_I_CELLS	Probar todos los input ante fallos tipo open
NET_REPS	Contar apariciones de la net
NETS	Recorrer todas las redes
NET_GEN	Generar vectores STUCK AT y SHORT

16 - Contadores empleados

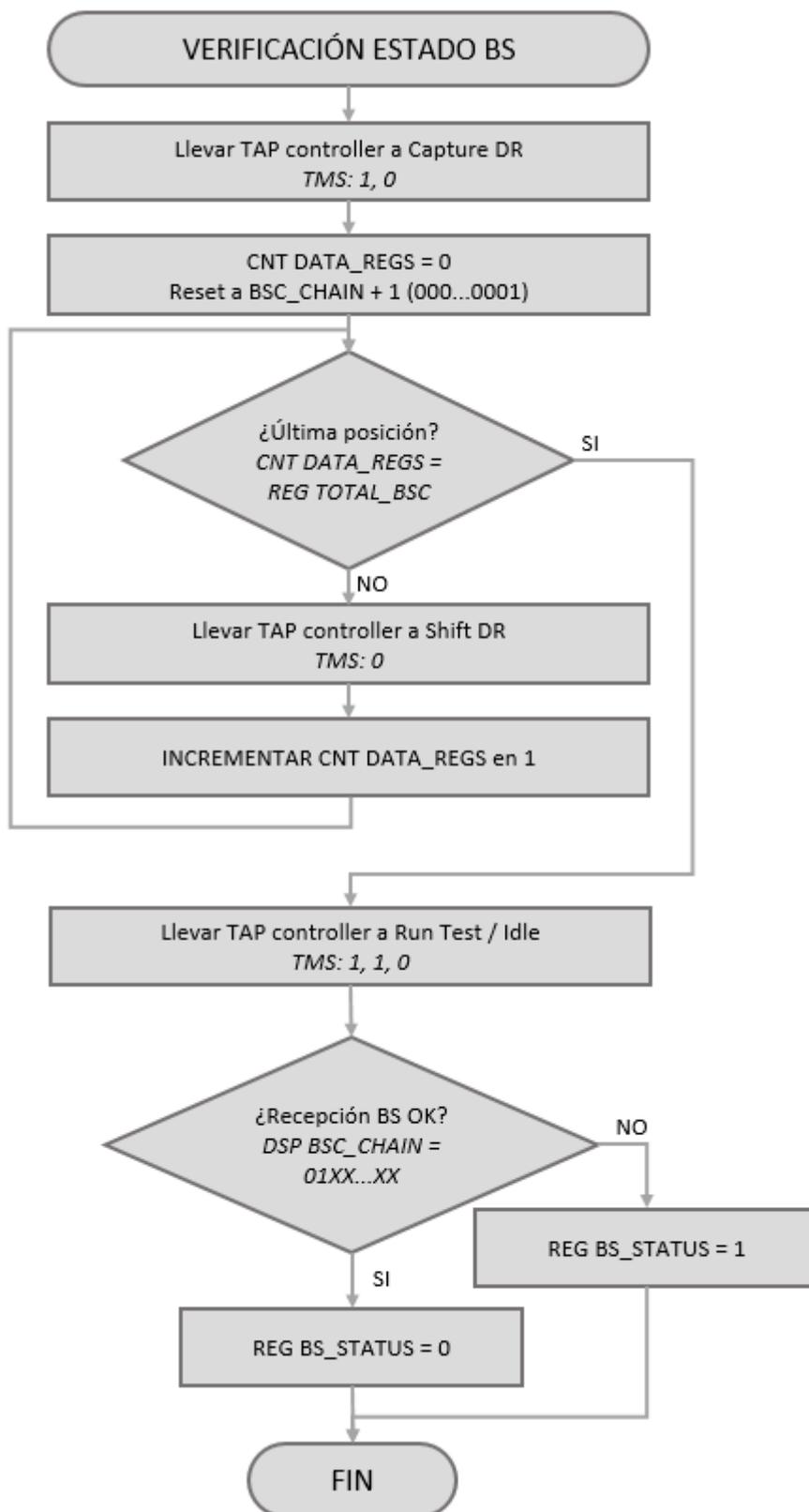
El flujograma empleado para la implementación del algoritmo es el siguiente:



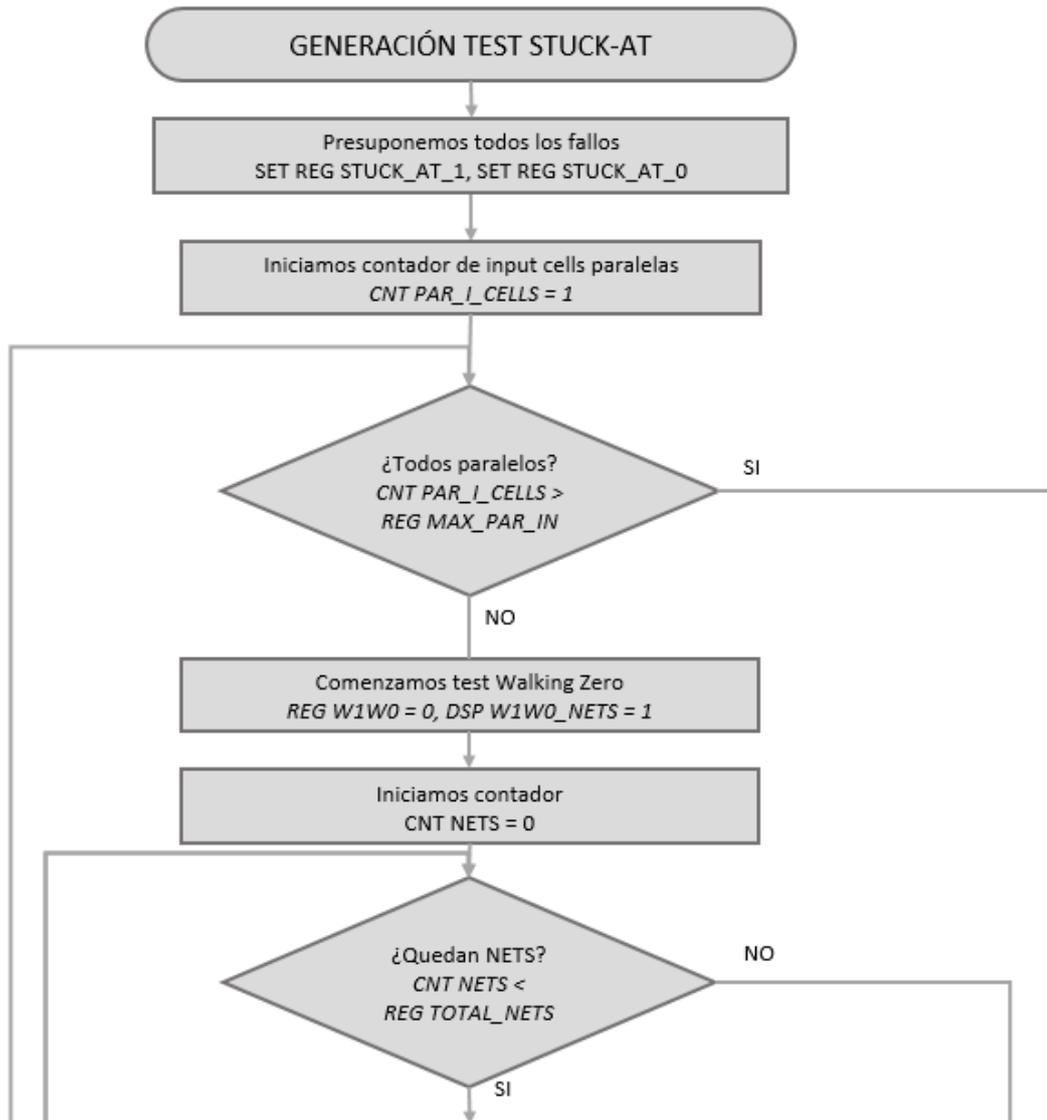
57 - Inicio del algoritmo



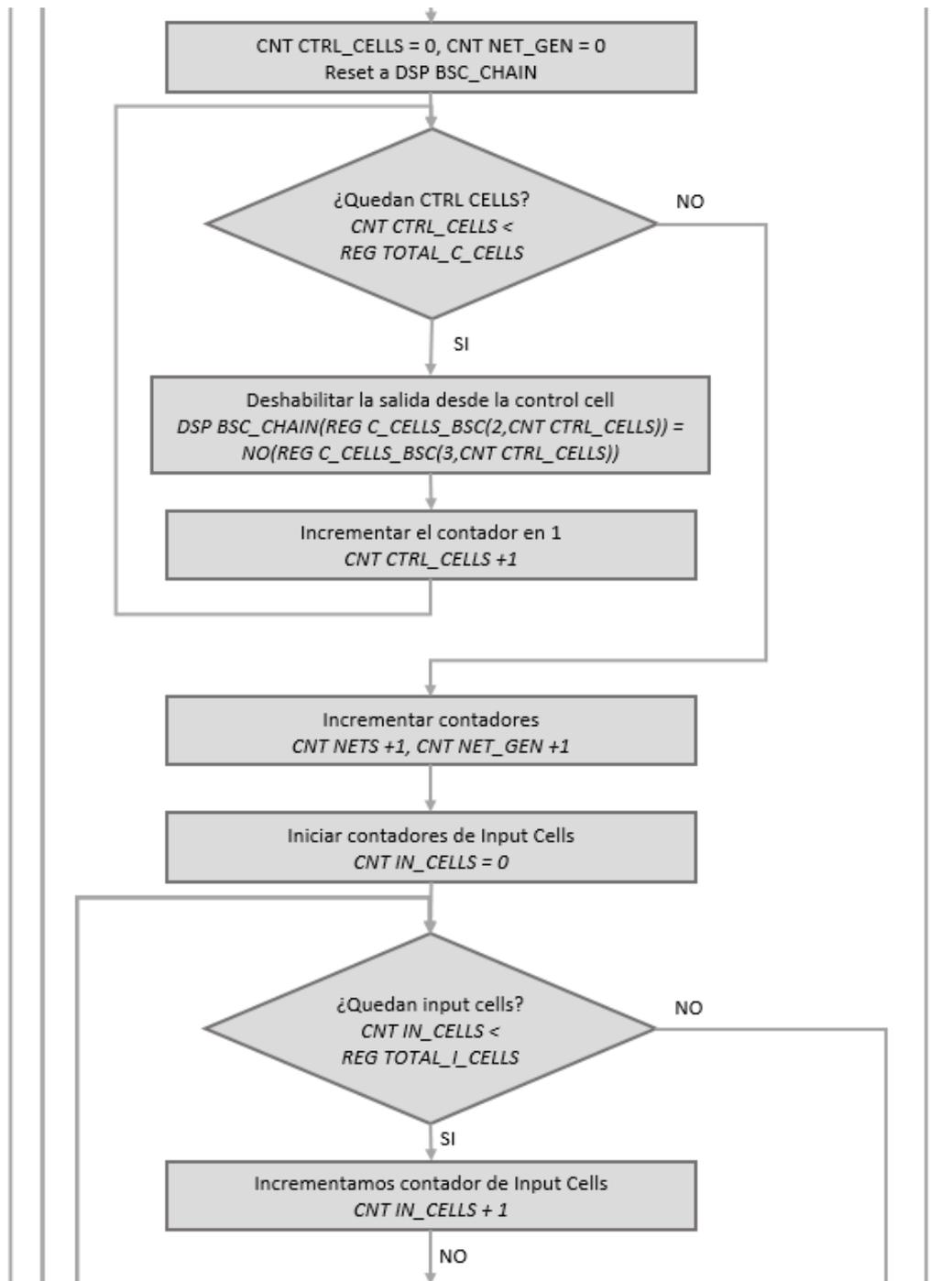
58 - Verificación del estado JTAG



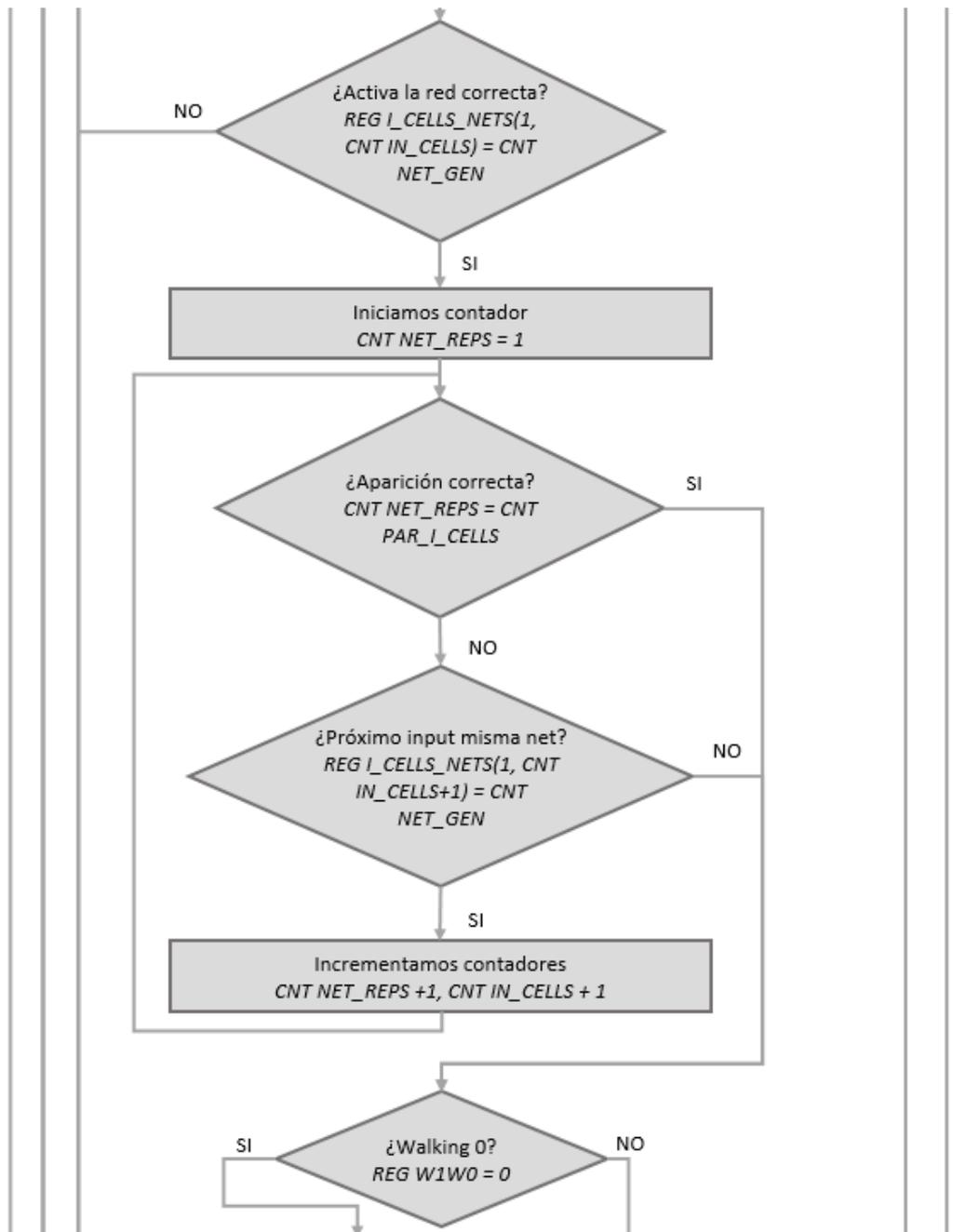
59 - Verificación del estado boundary-scan



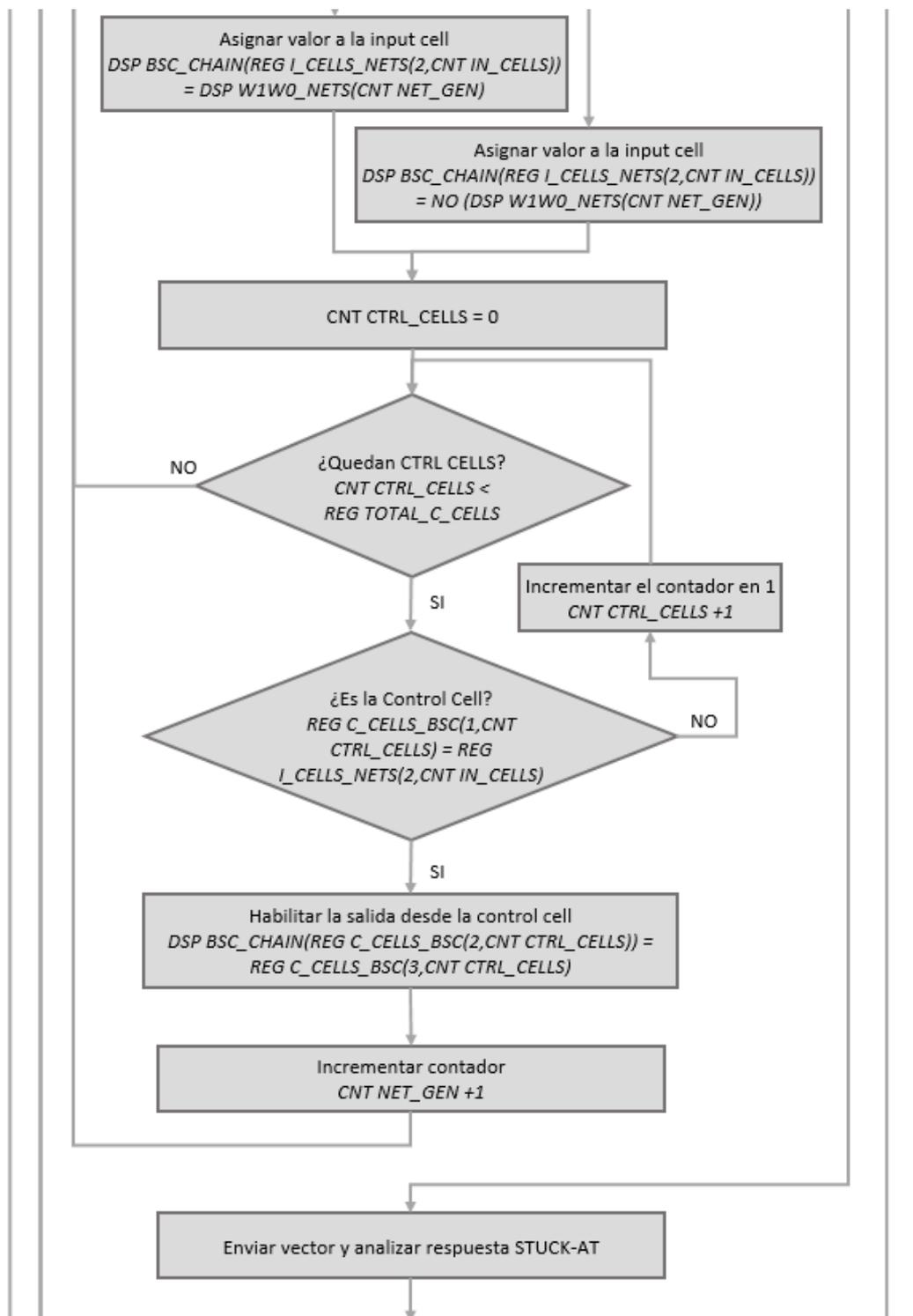
60 - Generación test stuck-at - parte 1



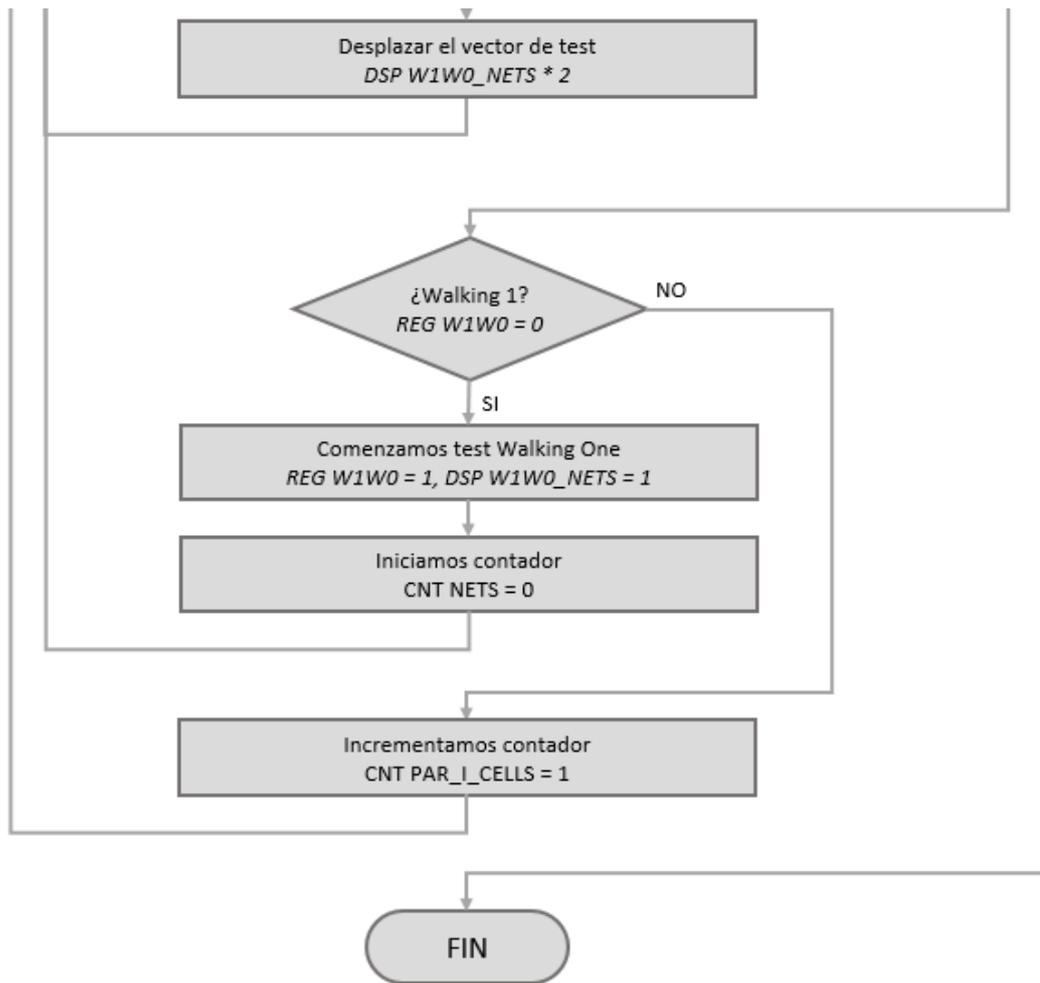
61 - Generación test stuck-at - parte 2



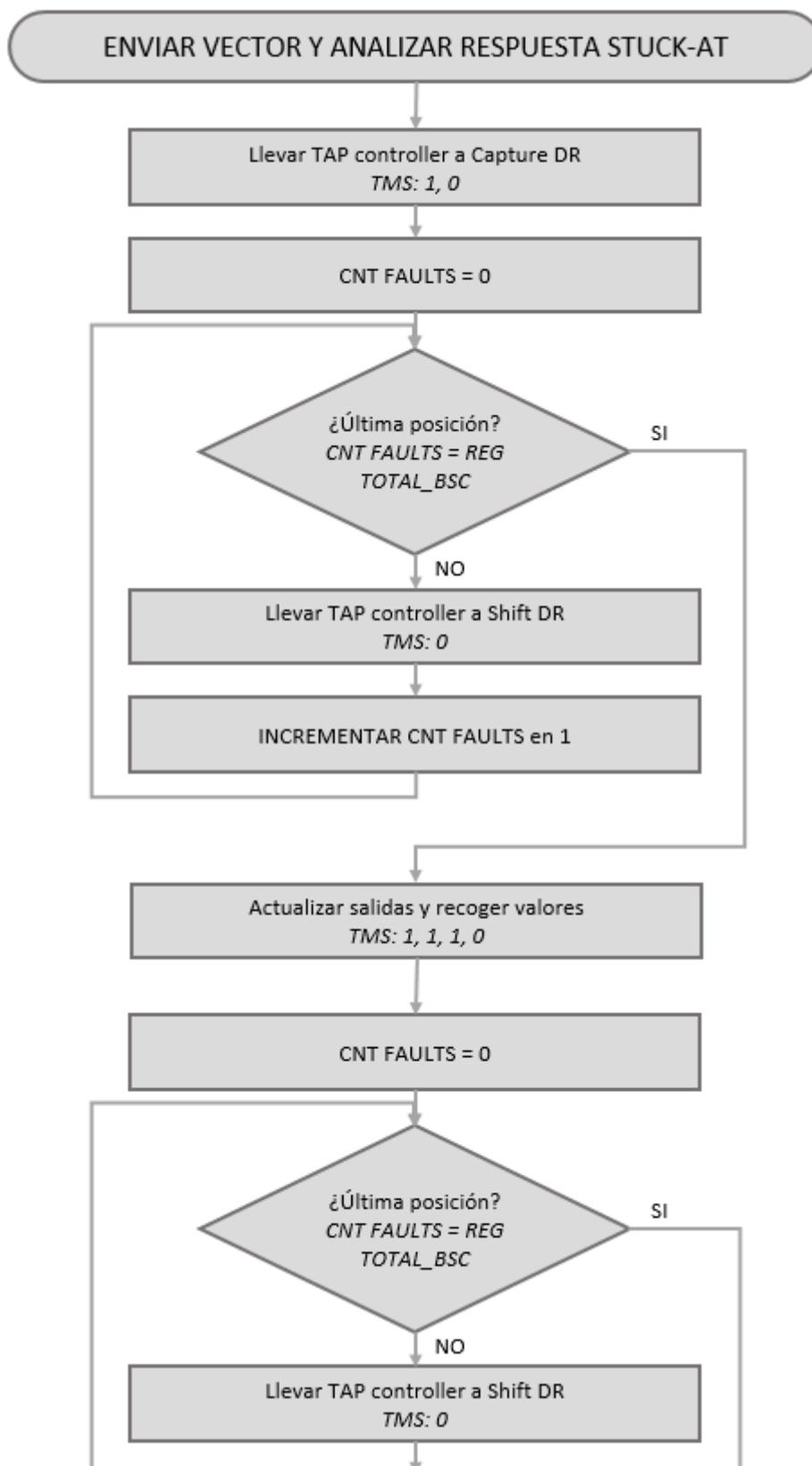
62 - Generación test stuck-at - parte 3



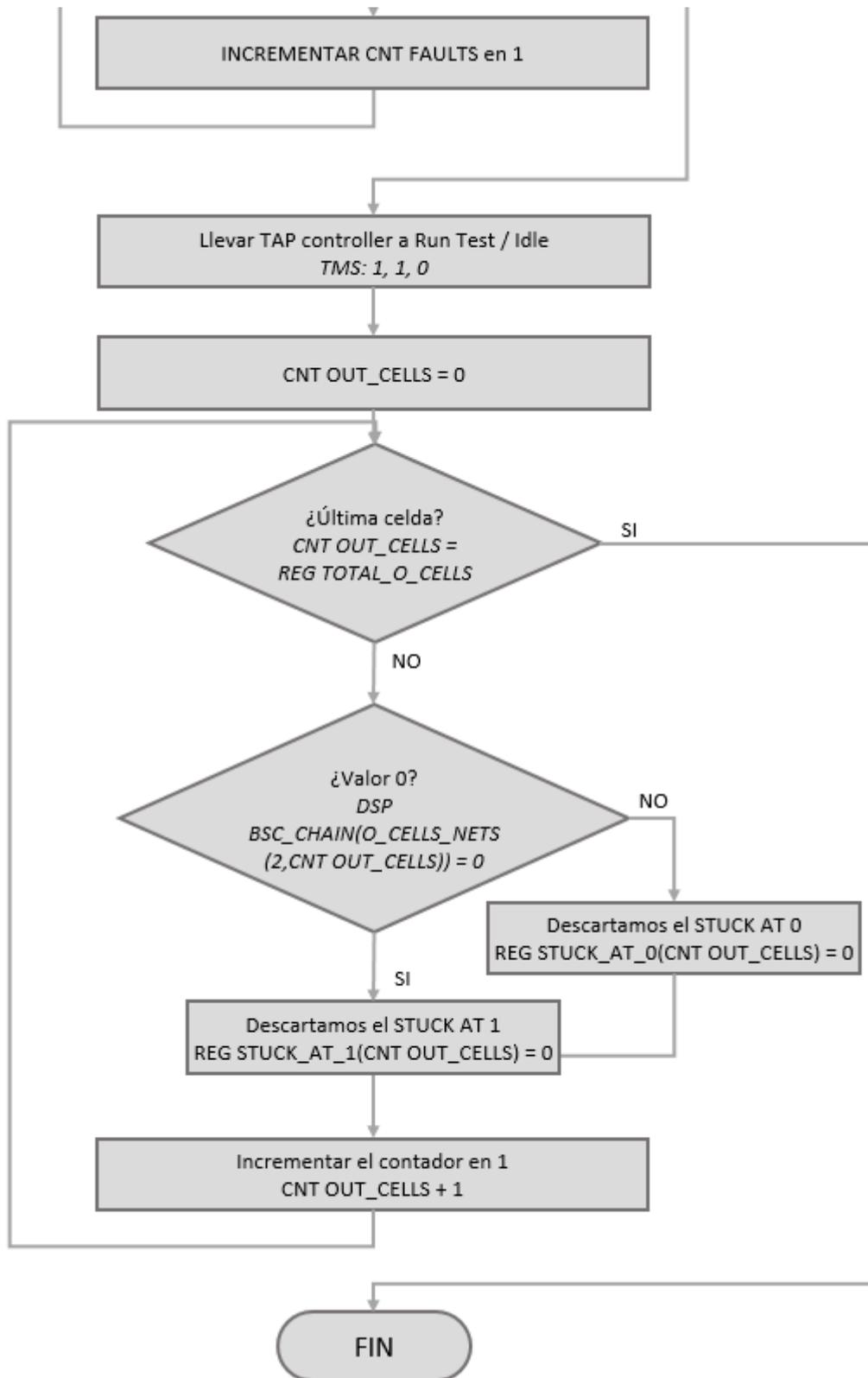
63 - Generación test stuck-at - parte 4



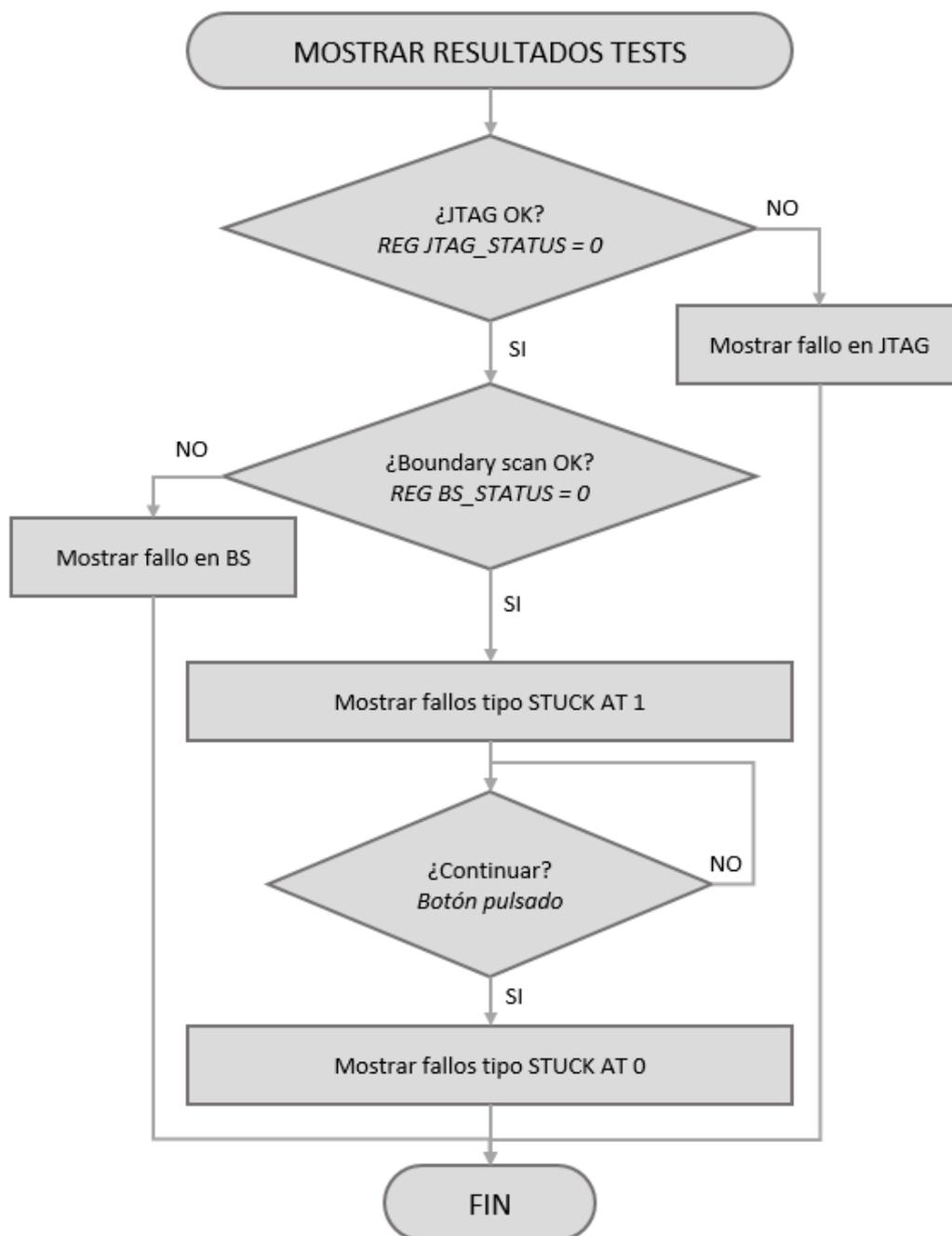
64 - Generación test stuck-at - parte 5



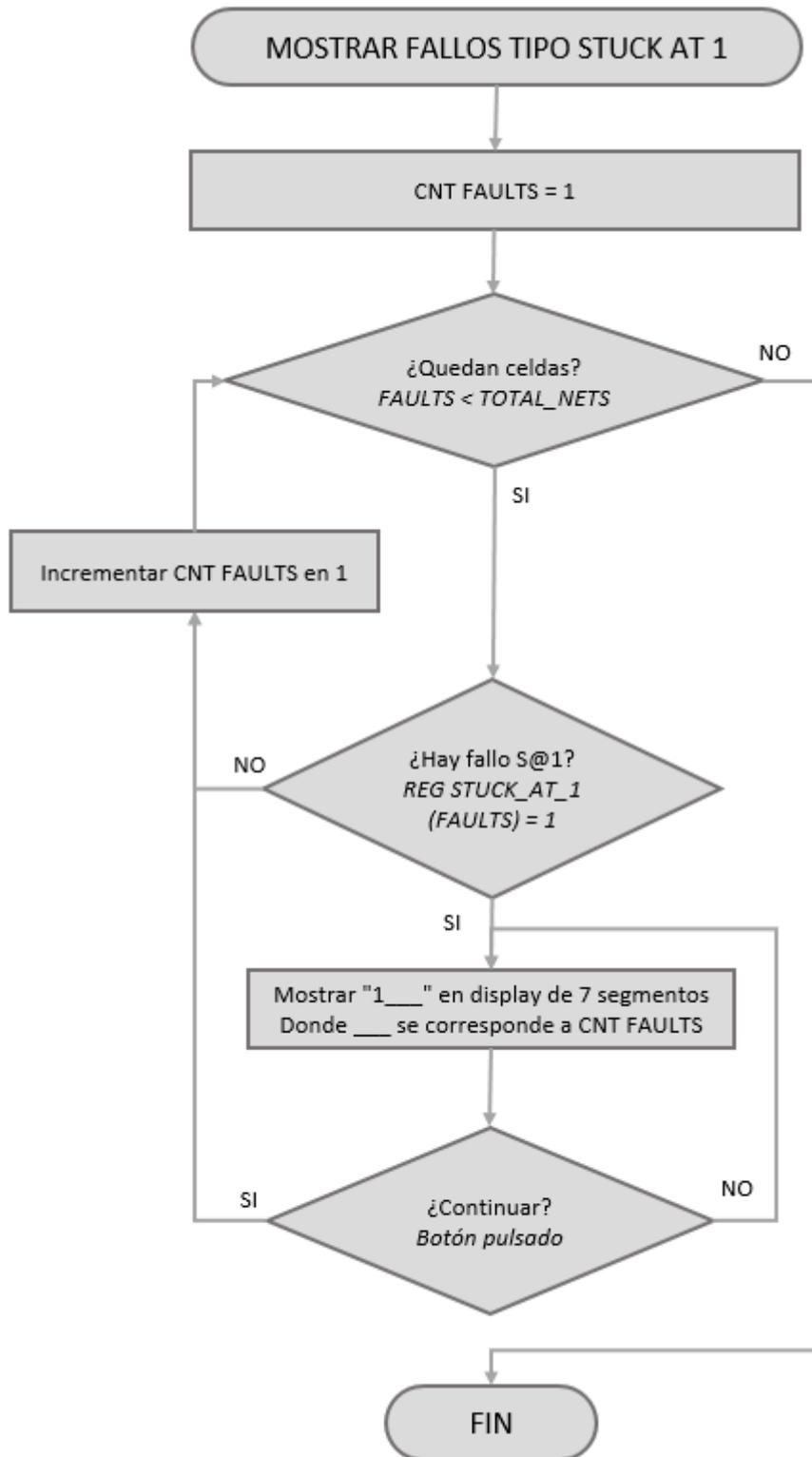
65 - Envío y análisis del vector de test - parte 1



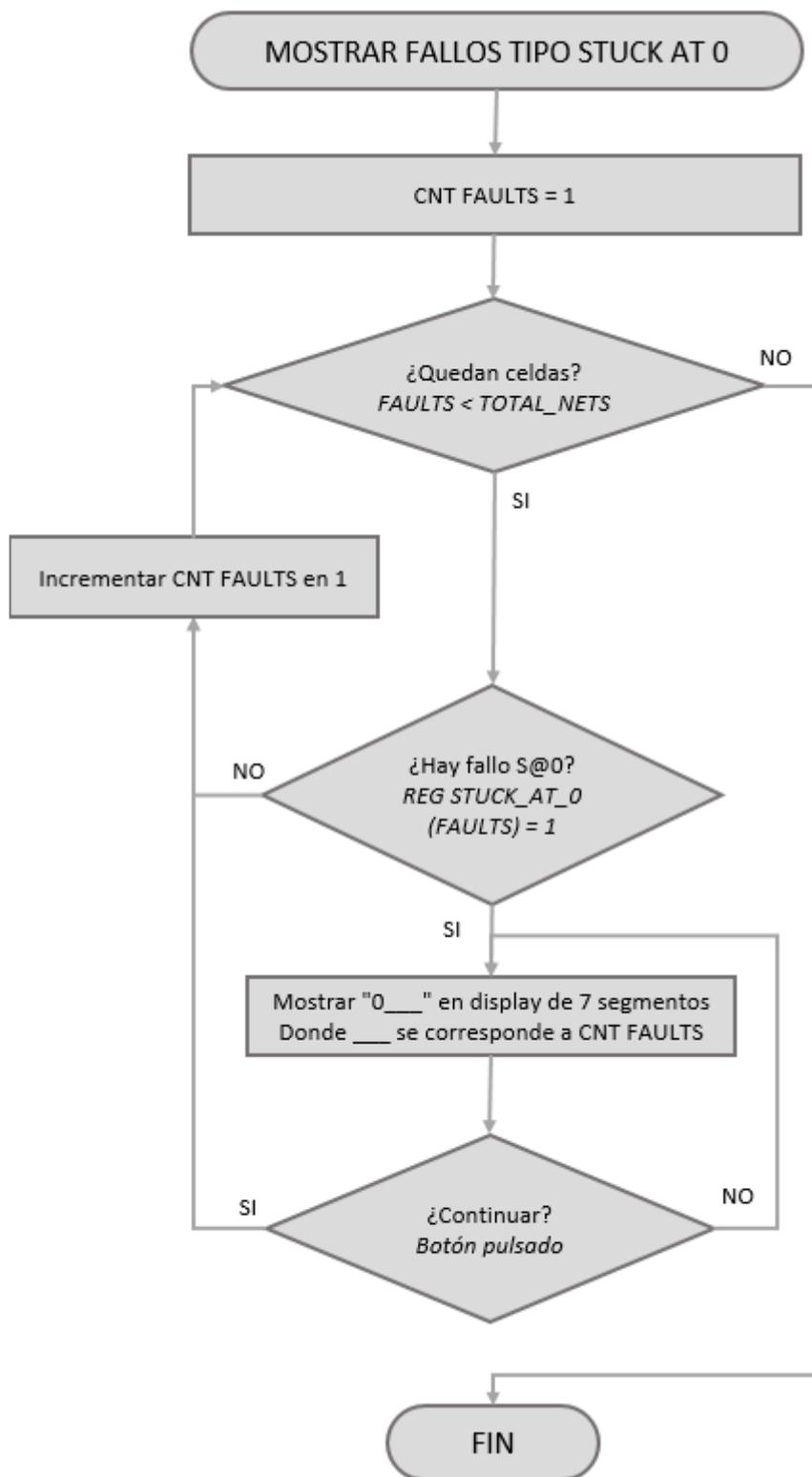
66 - Envío y análisis del vector de test - parte 2



67 - Mostrar resultados del test



68 - Mostrar fallos tipo stuck-at-one



69 - Mostrar fallos tipo stuck-at-zero

4.4 Resultados experimentales

Tras realizar la depuración del algoritmo aquí planteado, se hace necesario proceder a la verificación del mismo. Para ello se emplea la tarjeta diseñada en el capítulo 3.

El primer paso a realizar es comprobar la integridad de la cadena JTAG, encargada de comunicar los puertos de acceso para test. Esta verificación se realiza tomando dos enfoques distintos, en el primero de ellos no se conecta físicamente la tarjeta de test a la FPGA, en el segundo ésta sí se conecta, pero se retira el PIC32MX250. Estos test dan el mismo resultado, fallo en el puerto JTAG, un comportamiento correcto, pues no hay integridad de la señal entre TDI y TDO de la FPGA.

La detección de una cadena *boundary-scan* defectuosa no se puede realizar, sin embargo, sí se puede comprobar que se verifica la misma correctamente, para lo que se realiza un test con la placa conectada sin configurar fallo alguno. Esta prueba no arroja ningún defecto, es decir, el vector de fallos correspondientes a los *stuck-at-one* y *stuck-at-zero* no indica fallos, por lo que se conoce el estado correcto de la cadena *boundary-scan*.

El siguiente test a realizar es la detección de un fallo de tipo *stuck-at-one* en la patilla 11 del primer integrado, junto a un fallo de tipo *stuck-at-zero* en la patilla 12 del mismo integrado. Esto resulta en la correcta identificación del defecto, pues el vector de fallos *stuck-at-one* tiene activo el bit correspondiente a la primera red, mientras que el vector de fallos *stuck-at-zero* indica un fallo en la segunda.

Se continúa con test similares en los que se prueban de forma aislada las trece redes que tiene la placa de test, el comportamiento del algoritmo es el correcto, detectando exitosamente los diversos fallos configurados.

Finalmente se realiza una simulación sobre varias pistas con diferentes fallos de tipo *stuck-at-one* y *stuck-at-zero* mediante la configuración de la placa de test. El algoritmo presenta un funcionamiento correcto, detectando los defectos eléctricos presentes en cada caso.

5 Estudio económico

En este capítulo se exponen los costes de todo el trabajo fin de grado. Se detallarán los diversos tipos de costes: directos e indirectos.

5.1 Recursos empleados

En la elaboración de este trabajo se han empleado diversos recursos, se detallan a continuación:

- Ordenador empleado para el desarrollo del trabajo fin de grado: Surface Pro 2.
- Sistema operativo Windows 10 Pro 64 bits.
- Vivado Web Pack Edition 2016.3
- Proteus 8 Profesional.
- Paquete ofimático Microsoft Office 365 ProPlus.
- Plataforma de desarrollo Basys 3.
- Materiales químicos empleados para revelar la placa.
- Componentes empleados en la placa generadora de defectos eléctricos.
- Impresora para revelar la placa.
- Consumibles de oficina: folios, bolígrafos, rotuladores, lapiceros, tóner de impresora.

5.2 Costes directos

Los costes directos son imputables al propio desarrollo del trabajo. Se incluye, pero no se limita a la mano de obra, la amortización de equipos y programas, o los materiales empleados.

5.2.1 Costes de personal

El personal involucrado en el desarrollo de este trabajo fin de grado han sido 2 ingenieros. Uno de ellos se ha encargado de realizar todas las investigaciones oportunas, diseñar tanto placa como algoritmo e implementar este último. El otro ingeniero ha colaborado en la realización de la placa aportando algunas modificaciones a la placa de test, también se ha encargado de realizar el revelado y soldado.

Para calcular el coste, primero se ha de calcular el coste por hora. Para ello se toma un sueldo bruto de 30 000 € anuales, a lo que se ha de incorporar la cotización a la seguridad social. La cotización a la seguridad social supone un 35% del sueldo bruto, lo que se implica 10 500 € anuales. En total, el coste anual de un ingeniero serían 40 500 €.

Para poder calcular el precio por hora se han de conocer el total de días trabajados:

Año medio	365.25 días
Sábados y Domingos	104.36 días
Vacaciones efectivas	20.00 días
Festivos reconocidos	15.00 días
Días perdidos estimados	5.00 días
Total días efectivos estimados	220.89 días

17 - Días efectivos estimados

Suponiendo una jornada laboral de 8 horas, el coste por hora de un ingeniero es:

$$\frac{40500 \frac{\text{€}}{\text{año}}}{220.89 \frac{\text{días}}{\text{año}} * 8 \frac{\text{horas}}{\text{día}}} = 22.92 \frac{\text{€}}{\text{hora}}$$

A continuación, se muestra un desglose de las horas empleadas:

Estudio del problema	47 horas
Formación y documentación	93 horas
Estudio de soluciones	39 horas
Desarrollo de la solución	200 horas
Elaboración de la documentación	97 horas
Total de horas empleadas	476 horas

18 - Total de horas empleadas en el trabajo fin de grado

Por tanto, el coste total de personal alcanza una cantidad de:

$$22.92 \frac{\text{€}}{\text{hora}} * 476 \text{ horas} = 10\,909.92 \text{ €}$$

5.2.2 Costes de amortización de equipos y programas

En esta sección se incluye el costo de los diferentes equipos, sistemas operativos y programas empleados cuya vida útil es mayor a la del presente trabajo. Para calcular el coste total se emplea el tiempo de amortización, la vida útil estimada. El factor de amortización se calcula en base al total de años estimados de vida útil.

Material	Importe	Factor de amortización	Amortización anual
Microsoft Surface Pro 2 (Incluye SO Windows 10 Pro)	1200 €	0.20	240 €
Vivado WebPack 2016.3	0 €	1	0 €
Proteus 8 Professional	526 €	0.20	105.2 €
Paquete ofimático Microsoft Office 365 ProPlus	0 €	1	0 €
TOTAL	1726 €		345.20 €

19 - Amortización anual de los equipos y programas

Finalmente se calcula el coste total de las horas dedicadas al trabajo:

$$\frac{345.20 \frac{\text{€}}{\text{año}}}{1767.12 \frac{\text{horas}}{\text{año}}} * 476 \text{ horas} = 92.99 \text{ €}$$

5.2.3 Costes derivados de otros materiales

Aquí se incluyen los costes de todo tipo de consumibles empleados durante la elaboración del trabajo y de su documentación. También se incluye la Basys 3.

Material	Importe
Basys 3	149 €
Cable conexión Basys 3 a ordenador	5 €
Componentes placa de test	30.85 €
Compuestos químicos para revelado	4 €
Fotocopias	6.50 €
Material de oficina	11 €
Total	206.35 €

20 - Otros costes directos

5.2.4 Costes directos totales

Los costes directos totales, considerando las tres categorías indicadas, ascienden a un total de:

$$\text{Costes de personal} + \text{Costes de amortización} + \text{Costes derivados} =$$

$$10\ 909.92 \text{ €} + 92.99 \text{ €} + 206.35 \text{ €} = 11\ 209.26 \text{ €}$$

5.3 Costes indirectos

Los costes indirectos se corresponden a los gastos derivados de la actividad asociada al trabajo fin de grado pero que no son imputables directamente, esto se debe a que pueden estar involucrados en otros proyectos o procesos. A continuación se muestran estos costes:

Dirección y servicios administrativos	150 €
Consumo de electricidad	35 €
Consumo en desplazamientos	20 €
Total	205 €

21 - Costes indirectos

5.4 Costes totales

Para obtener el coste total de este trabajo fin de grado se han de sumar ambas categorías de gasto:

$$\textit{Coste total} = \textit{Coste directo} + \textit{Coste indirecto} =$$

$$11\,209.26 \text{ €} + 205 \text{ €} = 11\,414.26 \text{ €}$$

Como se observa, el coste final del trabajo es de 11414.26 €.

6 Conclusiones

La elaboración de este trabajo ha supuesto la realización de un test eléctrico que verifica los fallos de tipo *stuck-at-one* y *stuck-at-zero*. Para hacer esto posible se han valorado todas las posibles configuraciones que pueden tener las redes en las tarjetas de circuito impreso, incluyendo la presencia de las denominadas redes triestado.

Una de las tareas más considerables ha sido la correspondiente a la búsqueda de un algoritmo que permita no sólo la generación de vectores de test, sino la correcta identificación de los defectos eléctricos que pueden ocurrir. Como se ha expuesto, la búsqueda del algoritmo generador es exitosa, mas no se ha encontrado algoritmo de una sola etapa capaz de identificar de forma certera los defectos presentes en una tarjeta, por lo que ha sido necesario tomar un enfoque distinto.

En cuanto a la tarjeta desarrollada, ésta permite configurar no sólo los defectos analizados en el presente trabajo fin de grado, sino que pretende servir para trabajos futuros en los que sea necesario cubrir todos los defectos deterministas. Esto se debe al estudio realizado sobre los posibles defectos e interacciones entre dichos fallos.

La implementación se ha realizado en base al diagrama de flujo presentado en el apartado 4.3. El estudio, realización y análisis previo de este diagrama de flujo ha permitido reducir considerablemente la dificultad en el momento de realizar la programación VHDL. Pese a la dificultad que presenta, realizar esta implementación a bajo nivel permite alcanzar mejores resultados en cuanto a tiempo de test que otras soluciones que hacen uso de hardware genérico.

Tras la realización de este trabajo fin de grado se cuenta con una FPGA capaz de identificar los defectos eléctrico de tipo *stuck-at* presentes en la tarjeta diseñada.

6.1 Trabajo futuro

Durante la realización de este trabajo fin de grado se ha tenido presente que el mismo sirviese como base en cuanto a la detección de defectos eléctricos que afectan a la interconexión de pistas. Muestra de ello es la placa diseñada, que permite simular múltiples fallos a la vez sobre una pista, como cortocircuitos y pistas abiertas.

El esqueleto del algoritmo desarrollado también se ha hecho teniendo presente que el mismo permita la detección de los defectos eléctricos no incluidos en el presente trabajo, como la presencia de pistas abiertas o cortocircuitos entre múltiples pistas. Las modificaciones que sería preciso realizar al algoritmo se encuentran detalladas en el anejo 14, donde se detallan en un diagrama de flujo los pasos necesarios para la detección de este tipo de fallos que se apartan del alcance de este trabajo fin de grado.

Dejando a un lado la detección de los diversos fallos, otra vía de desarrollo es la gestión de estos test de forma remota, de tal manera que el dispositivo sobre el que se realiza la implementación notifique el fallo presente. Se trata de un test que puede realizarse durante el tiempo ocioso con el que cuente el dispositivo, facilitándose así el mantenimiento y reparación del mismo.

7 Referencias

Las referencias consultadas para el desarrollo de este trabajo fin de grado han sido las siguientes:

- [1] The Institute of Electrical and Electronics Engineers, Inc, «IEEE Standard Test Access Port and Boundary-Scan Architecture». 23-jul-2001.
- [2] Najmi Jarwala and Chi W. Yau, «A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects», 1989.
- [3] Abu Hassan, Janusz Rajski, y Vinod K. Agarwal, «Testing and Diagnosis of Interconnects using Boundary Scan Architecture», 1988.
- [4] Pankaj Kumar, R. K. Sharma, D. K. Sharma, y B. K. Kaushik, «A Novel Method for Diagnosis of Board Level Interconnect Faults Using Boundary Scan». 2010.
- [5] Corelis, «Boundary Scan». [En línea]. Disponible en: http://www.corelis.com/education/Boundary-Scan_Tutorial.htm. [Accedido: 09-feb-2017].
- [6] Yongjoon Kim, Hyun-don Kim, y Sungho Kang, «A New Maximal Diagnosis Algorithm for Interconnect Test», may 2014.
- [7] W. K. Kautz, «Testing of Faults in Wiring Interconnects», abr. 1974.
- [8] P. Goel y M. T. McMahon, «Electronic Chip-in Place Test», jun. 1982.
- [9] P. T. Wagner, «Interconnect Testing with Boundary Scan», 1987.
- [10] Abu S. M. Hassan, Vinod K. Agarwal, Benoir Nadeau-Dostie, y Janusz Rajski, «BIST and Interconnect Testing with Boundary Scan». 10-oct-1992.
- [11] «Basys 3 Artix-7 FPGA Trainer Board: Recommended for Introductory Users», *Digilent*. [En línea]. Disponible en:

- <http://store.digilentinc.com/basys-3-artix-7-fpga-trainer-board-recommended-for-introductory-users/>. [Accedido: 09-feb-2017].
- [12] Digilent, «Basys 3™ FPGA Board Reference Manual». [En línea]. Disponible en: https://reference.digilentinc.com/_media/basys3:basys3_rm.pdf. [Accedido: 09-feb-2017].
- [13] «Artix-7 FPGA Family». [En línea]. Disponible en: <https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html>. [Accedido: 09-feb-2017].
- [14] Peter Clarke, «Xilinx, ASIC Vendors Talk Licensing». 22-jun-2001.
- [15] «Vivado Design Suite Evaluation and WebPACK». [En línea]. Disponible en: <https://www.xilinx.com/products/design-tools/vivado/vivado-webpack.html>. [Accedido: 09-feb-2017].
- [16] Digilent, «Digilent Pmod Interface Specification». [En línea]. Disponible en: https://www.digilentinc.com/Pmods/Digilent-Pmod_%20Interface_Specification.pdf. [Accedido: 09-feb-2017].
- [17] Pavankumar Banakar, Rinku P Mathew, y Cypress Semiconductor, «Protecting your low voltage electronic devices from electrical overstress». [En línea]. Disponible en: <http://www.embedded.com/design/prototyping-and-development/4423709/Protecting-your-low-voltage-electronic-devices-from-electrical-overstress>. [Accedido: 09-feb-2017].
- [18] Littelfuse, «PolyZen Devices for EOS Protection in LED Applications», jul-2016. [En línea]. Disponible en: http://www.littelfuse.com/~/_media/electronics/application_notes/littelfuse_polyzen_devices_for_eos_protection_in_led_application_note.pdf. [Accedido: 09-feb-2017].
- [19] «ATmega324A - Atmel-42714-ATmega324A_Datasheet.pdf». [En línea]. Disponible en: http://www.atmel.com/Images/Atmel-42714-ATmega324A_Datasheet.pdf. [Accedido: 09-feb-2017].

- [20] «Atmel AVR ATmega162 datasheet - Atmel-2513-8-bit-AVR-Microntrroller-ATmega162_Datasheet.pdf». [En línea]. Disponible en: http://www.atmel.com/Images/Atmel-2513-8-bit-AVR-Microntrroller-ATmega162_Datasheet.pdf. [Accedido: 09-feb-2017].
- [21] «ATMEGA162-16PU ATMEL, Microcontrolador de 8 Bits, Baja Potencia, Alto Rendimiento, ATmega, 16 MHz, 16 KB, 1 KB, 40 Pines | Farnell element14 España». [En línea]. Disponible en: <http://es.farnell.com/atmel/atmega162-16pu/mcu-8bit-atmega-16mhz-dip-40/dp/9171169>. [Accedido: 09-feb-2017].
- [22] «ATMEGA324A-PU ATMEL, Microcontrolador de 8 Bits, Baja Potencia, Alto Rendimiento, ATmega, 20 MHz, 32 KB, 2 KB, 40 Pines | Farnell element14 España». [En línea]. Disponible en: <http://es.farnell.com/atmel/atmega324a-pu/mcu-8bit-avr-32-k-flash-40pdip/dp/1841611>. [Accedido: 09-feb-2017].
- [23] «SN54LVTH18502A, SN54LVTH182502A, SN74LVTH18502A, SN74LVTH182502A (Rev. C) - sn74lvth18502a.pdf». [En línea]. Disponible en: <http://www.ti.com/lit/ds/symlink/sn74lvth18502a.pdf>. [Accedido: 09-feb-2017].
- [24] «3.3-V ABT Scan Test Devices With 20-Bit Universal Bus Transceivers (Rev. B) - sn74lvth18504a.pdf». [En línea]. Disponible en: <http://www.ti.com/lit/ds/symlink/sn74lvth18504a.pdf>. [Accedido: 09-feb-2017].
- [25] «SN74LVTH18502APM TEXAS INSTRUMENTS, Dispositivo de Prueba de Exploración, 9 Canales, 2.7 V a 3.6 V, LQFP-64 | Farnell element14 España». [En línea]. Disponible en: <http://es.farnell.com/texas-instruments/sn74lvth18502apm/ic-scan-test-device-smd-lqfp64/dp/1236456?ost=SN74LVTH18502APM&selectedCategoryId=&categoryNameResp=Todas%2Blas%2Bcategor%25C3%25ADas&searchView=table&isrcfnonsku=false>. [Accedido: 09-feb-2017].
- [26] «Scan Test Devices With Octal Buffers (Rev. E) - sn74bct8244a.pdf». [En línea]. Disponible en: <http://www.ti.com/lit/ds/symlink/sn74bct8244a.pdf>. [Accedido: 09-feb-2017].

- [27] «Scan Test Devices With Octal Inverting Buffers (Rev. E) - sn74bct8240a.pdf». [En línea]. Disponible en: <http://www.ti.com/lit/ds/symlink/sn74bct8240a.pdf>. [Accedido: 09-feb-2017].
- [28] «SN74BCT8244ADW | Búfer, controlador de línea, SN74BCT8244ADW, BCT, 8 bits 3-State No Inversión SOIC 24 pines | Texas Instruments». [En línea]. Disponible en: <http://es.rs-online.com/web/p/combinaciones-de-driver-de-linea-y-bufer/6631650/>. [Accedido: 09-feb-2017].
- [29] «PIC32MX1XX/2XX 28/36/44-pin Family Data Sheet - 60001168J.pdf». [En línea]. Disponible en: <http://ww1.microchip.com/downloads/en/DeviceDoc/60001168J.pdf>. [Accedido: 09-feb-2017].
- [30] «PIC32MX110F016B-I/SP MICROCHIP, Microcontrolador PIC/DSPIC, Interfaz de Audio y Gráficos, PIC32, 32bit, 40 MHz, 16 KB, 4 KB | Farnell element14 España». [En línea]. Disponible en: <http://es.farnell.com/microchip/pic32mx110f016b-i-sp/mcu-32bit-pic32-40mhz-sdip-28/dp/2467671?exaMfpn=true&categoryId=&searchRef=SearchLookAhead&searchView=table&isrcfnonsku=false>. [Accedido: 09-feb-2017].
- [31] «PIC32MX250F128B-50I/SP MICROCHIP, Microcontrolador PIC/DSPIC, Interfaz de Audio y Gráficos, PIC32, 32bit, 50 MHz, 128 KB, 32 KB | Farnell element14 España». [En línea]. Disponible en: <http://es.farnell.com/microchip/pic32mx250f128b-50i-sp/mcu-32bit-pic32-50mhz-sdip-28/dp/2313768?ost=2313768&selectedCategoryId=&categoryNameResp=Todas%2Blas%2Bcategor%25C3%25ADas&searchView=table&isrcfnonsku=false>. [Accedido: 09-feb-2017].
- [32] «Farnell element14 España - distribuidor de componentes electrónicos». [En línea]. Disponible en: <http://es.farnell.com/>. [Accedido: 09-feb-2017].

8 Anejos

Los anejos se encuentran disponibles en el documento adjunto, no obstante, se indica aquí el listado de los mismos:

1. Basys 3 – Schematics.
2. Basys 3 – Reference manual.
3. Datasheet PIC32MX110F016B.
4. Datasheet PIC32MX250F128B.
5. Datasheet Puerta lógica AND - CD4082B.
6. Datasheet Puerta lógica OR - CD4072B.
7. BSDL_Model_of_PIC32MX110F016B.
8. BSDL_Model_of_PIC32MX250F128B.
9. Digilent Pmod Interface Specification.
10. Placa defectos eléctricos - Schematics.
11. Placa defectos eléctricos - Layout.
12. Diagrama de flujo de la implementación realizada.
13. Programación en VHDL.
14. Propuesta de flujograma con mayor cobertura de fallos.