



MAESTRÍA EN  
SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

**Sistema ferroviario de enclavamiento  
electrónico en FPGA**

**Autor:**

Esp. Ing. Martín Nicolás Menéndez

Director:  
Dr. Ing. Ariel Lutenberg (CONICET-FIUBA)  
Codirector:  
Mg. Ing. Facundo Larosa ([UTN-FRBA](#))

Jurados:  
Mg. Ing. Nicolás Dassieu Blanchet (FIUBA)  
Esp. Ing. Pedro Martos (FIUBA)  
Ing. Nicolás Álvarez (UNSAM, FIUBA)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires,  
entre marzo de 2019 y abril de 2020.*



MAESTRÍA EN  
SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

**Sistema ferroviario de enclavamiento  
electrónico en FPGA**

**Autor:**

Esp. Ing. Martín Nicolás Menéndez

Director:  
Dr. Ing. Ariel Lutenberg (CONICET-FIUBA)  
Codirector:  
Mg. Ing. Facundo Larosa ([UTN-FRH](#))

Jurados:  
Mg. Ing. Nicolás Dassieu Blanchet (FIUBA)  
Esp. Ing. Pedro Martos (FIUBA)  
Ing. Nicolás Alvarez (UNSAM, FIUBA)

*Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires,  
entre marzo de 2019 y abril de 2020.*

III

III

### *Resumen*

En base a un pedido de Trenes Argentinos, se diseñó un sistema de enclavamiento electrónico. Un enclavamiento es un sistema ferroviario de alto costo y complejidad, que controla automáticamente que las máquinas de cambios y las señales se gestionen de forma segura, evitando choques y descarrilamientos de trenes.

Para el desarrollo de este trabajo resultó necesario aplicar conocimientos de lenguajes de descripción de hardware para la implementación del sistema sobre una plataforma de lógica programable en FPGA. Además, se trabajó con control de versiones y se aplicó el modelo de máquinas de estados finitos para la lógica del sistema. Se aplicaron estrategias de testing para validar los módulos. Finalmente, fueron fundamentales los conocimientos de sistemas críticos y ferroviarios.

### *Resumen*

En base a un pedido de Trenes Argentinos, se diseñó un sistema de enclavamiento electrónico. Se trata de un sistema ferroviario de alto costo y complejidad, que controla automáticamente que las máquinas de cambios y las señales se gestionen de forma segura, evitando choques y descarrilamientos de trenes.

Para el desarrollo de este trabajo resultó necesario aplicar conocimientos de lenguajes de descripción de hardware para la implementación del sistema sobre una plataforma de lógica programable en FPGA. Además, se trabajó con control de versiones y se aplicó el modelo de máquinas de estados finitos para la lógica del sistema. Se aplicaron estrategias de testing para validar los módulos. Finalmente, fueron fundamentales los conocimientos de sistemas críticos y ferroviarios, aprendidos durante la Maestría.

V

## *Agradecimientos*

Quisiera agradecer a mi familia, que siempre me ha apoyado a lo largo de todos mis estudios de forma incondicional. Espero que se sientan tan orgullosos de los resultados como yo de haber puesto tanto esfuerzo en la realización de mi carrera.

A mi director, el Dr. Ing. Ariel Lutenberg, que con una enorme paciencia siempre me ha dado incontables oportunidades para demostrar mi capacidad. Sus inagotables ganas de que el país crezca tecnológicamente son muy inspiradoras y los proyectos en los que me ha confiado la participación me enriquecen enormemente.

A mi codirector, el Mg. Ing. Facundo Larosa, que dispuso muchas de mis dudas y aportó una metodología de trabajo que supo dar sus frutos.

Al Ing. Nicolás Álvarez, que desde el inicio del proyecto supo guiarme en los apartados técnicos del mundo de las FPGAs y su dedicación al proyecto fue crucial cuando se requería una experiencia que me excedía.

Al CONICET-GICSAFe por su gran camaradería y solidaridad a la hora de compartir sus saberes, tanto de la profesión como de la vida diaria. En especial al Dr. Ing. Pablo Gomez y al Ing. Adrián Laiuppa, ya que conversando con ellos fuera del trabajo diario he aprendido enormemente.

V

## *Agradecimientos*

Quisiera agradecer a mi familia, que siempre me ha apoyado a lo largo de todos mis estudios de forma incondicional. Espero que se sientan tan orgullosos de los resultados como yo de haber puesto tanto esfuerzo en la realización de mi carrera.

A mi director, el Dr. Ing. Ariel Lutenberg, que con una enorme paciencia siempre me ha dado incontables oportunidades para demostrar mi capacidad. Sus inagotables ganas de que el país crezca tecnológicamente son muy inspiradoras y los proyectos en los que me ha confiado la participación me enriquecen enormemente.

A mi codirector, el Mg. Ing. Facundo Larosa, que dispuso muchas de mis dudas y aportó una metodología de trabajo que supo dar sus frutos.

Al Ing. Nicolás Álvarez, que desde el inicio del proyecto supo guiarme en los apartados técnicos del mundo de las FPGAs y su dedicación al proyecto fue crucial cuando se requería una experiencia que me excedía.

Al CONICET-GICSAFe por su gran camaradería y solidaridad a la hora de compartir sus saberes, tanto de la profesión como de la vida diaria. En especial al Dr. Ing. Pablo Gomez y al Ing. Adrián Laiuppa, ya que conversando con ellos fuera del trabajo diario he aprendido enormemente.

## Índice general

Resumen	III
<b>1. Introducción General</b>	
1.1. Contexto y motivación . . . . .	1
1.2. Propuesta de solución . . . . .	2
1.3. Elementos del señalamiento ferroviario . . . . .	4
1.3.1. Vías . . . . .	4
1.3.2. Semáforos ferroviarios . . . . .	5
1.3.3. Circuito de vías . . . . .	6
1.3.4. Pasos a nivel . . . . .	7
1.3.5. Máquina de cambios . . . . .	8
1.4. Sistema de enclavamientos . . . . .	9
1.5. Tipos de enclavamientos . . . . .	10
1.5.1. Enclavamientos mecánicos . . . . .	10
1.5.2. Enclavamientos electromecánicos . . . . .	11
1.5.3. Enclavamientos electrónicos . . . . .	12
1.6. Objetivos . . . . .	13
<b>2. Introducción Específica</b>	15
2.1. Topologías típicas . . . . .	15
2.1.1. Bypass . . . . .	15
2.1.2. Estación . . . . .	16
2.1.3. Hub . . . . .	16
2.1.4. Terminal . . . . .	17
2.2. Enfoque funcional . . . . .	18
2.2.1. Modelo del sistema . . . . .	19
2.2.2. Flujo de trabajo . . . . .	20
2.2.3. Escalabilidad de la estrategia . . . . .	21
2.3. Enfoque geográfico . . . . .	22
2.3.1. Análisis de grafos . . . . .	22
2.3.2. Flujo de trabajo . . . . .	23
2.3.3. Escalabilidad de la estrategia . . . . .	24
<b>3. Diseño e Implementación</b>	27
3.1. Consideraciones generales . . . . .	27
3.2. Plataforma utilizada . . . . .	28
3.3. Análisis de la red ferroviaria y generación automática del código . . . . .	29
3.4. Módulo de nodos . . . . .	30
3.5. Módulo de la máquina de cambios . . . . .	30
3.6. Módulos de adaptación a enclavamiento . . . . .	31
3.6.1. Módulo separador . . . . .	32
3.6.2. Módulo mediador . . . . .	33
3.7. Módulos de procesamiento de tramas . . . . .	33
3.7.1. Módulo detector . . . . .	33
3.7.2. Módulo registro . . . . .	35
3.7.3. Módulo selector . . . . .	36
3.8. Módulo de comunicación UART . . . . .	37
3.9. Interfaz de comunicación Python . . . . .	38
<b>4. Ensayos y Resultados</b>	41
4.1. Validación de grafos . . . . .	41
4.2. Validación del nodo . . . . .	41
4.2.1. Testbench del módulo nodo . . . . .	41
4.2.2. Resultados obtenidos . . . . .	44
4.3. Validación de la máquina de cambios . . . . .	45
4.3.1. Testbench del módulo de la máquina de cambios . . . . .	45
4.3.2. Resultados obtenidos . . . . .	49
4.4. Validación de la UART . . . . .	49
4.4.1. Testbench del módulo UART . . . . .	49
4.4.2. Resultados obtenidos . . . . .	54
4.5. Validación del detector . . . . .	54
4.5.1. Testbench del módulo detector . . . . .	55
4.5.2. Resultados obtenidos . . . . .	59
4.6. Validación del enclavamiento . . . . .	60
4.6.1. Testbench del módulo enclavamiento . . . . .	60
4.6.2. Resultados obtenidos . . . . .	64
4.7. Validación del registro . . . . .	65
4.7.1. Testbench del módulo registro . . . . .	65
4.7.2. Resultados obtenidos . . . . .	67
4.8. Validación del selector . . . . .	67
4.8.1. Testbench del módulo selector . . . . .	68
4.8.2. Resultados obtenidos . . . . .	70
<b>5. Conclusiones</b>	73
5.1. Resultados obtenidos . . . . .	73
5.2. Próximos pasos . . . . .	74
<b>Bibliografía</b>	75

## Índice general

Resumen	III
<b>1. Introducción General</b>	1
1.1. Contexto y motivación . . . . .	1
1.2. Propuesta de solución . . . . .	2
1.3. Elementos del señalamiento ferroviario . . . . .	6
1.3.1. Vías . . . . .	6
1.3.2. Semáforos ferroviarios . . . . .	6
1.3.3. Circuito de vías . . . . .	8
1.3.4. Pasos a nivel . . . . .	9
1.3.5. Máquina de cambios . . . . .	9
1.4. Sistema de enclavamientos . . . . .	11
1.5. Tipos de enclavamientos . . . . .	12
1.5.1. Enclavamientos mecánicos . . . . .	12
1.5.2. Enclavamientos electromecánicos . . . . .	12
1.5.3. Enclavamientos electrónicos . . . . .	13
1.6. Objetivos . . . . .	14
<b>2. Introducción Específica</b>	15
2.1. Topologías típicas . . . . .	15
2.1.1. Bypass . . . . .	15
2.1.2. Estación . . . . .	16
2.1.3. Hub . . . . .	16
2.1.4. Terminal . . . . .	17
2.2. Estrategias de resolución . . . . .	18
2.3. Enfoque funcional . . . . .	18
2.3.1. Tabla de enclavamientos . . . . .	18
2.3.2. Modelo del sistema . . . . .	20
2.3.3. Flujo de trabajo . . . . .	21
2.3.4. Escalabilidad de la estrategia . . . . .	21
2.4. Enfoque geográfico . . . . .	22
2.4.1. Análisis de grafos . . . . .	23
2.4.2. Flujo de trabajo . . . . .	24
2.4.3. Escalabilidad de la estrategia . . . . .	25
2.5. Consideraciones generales . . . . .	26
2.6. Plataforma utilizada . . . . .	27
<b>3. Diseño e Implementación</b>	29
3.1. Analizador de redes ferroviarias . . . . .	29
3.2. Módulo de nodos . . . . .	31
3.3. Módulo de la máquina de cambios . . . . .	32
3.4. Módulos de adaptación a enclavamiento . . . . .	33
3.4.1. Módulo separador . . . . .	34
3.4.2. Módulo mediador . . . . .	35
3.5. Módulos de procesamiento de tramas . . . . .	36
3.5.1. Módulo detector . . . . .	36
3.5.2. Módulo registro . . . . .	37
3.5.3. Módulo selector . . . . .	39
3.6. Módulo de comunicación UART . . . . .	40
3.7. Interfaz de comunicación Python . . . . .	41
<b>4. Ensayos y Resultados</b>	43
4.1. Validación de grafos . . . . .	43
4.2. Validación del nodo . . . . .	43
4.2.1. Testbench del módulo nodo . . . . .	43
4.2.2. Resultados obtenidos . . . . .	46
4.3. Validación de la máquina de cambios . . . . .	47
4.3.1. Testbench del módulo de la máquina de cambios . . . . .	47
4.3.2. Resultados obtenidos . . . . .	51
4.4. Validación de la UART . . . . .	51
4.4.1. Testbench del módulo UART . . . . .	51
4.4.2. Resultados obtenidos . . . . .	56
4.5. Validación del detector . . . . .	56
4.5.1. Testbench del módulo detector . . . . .	57
4.5.2. Resultados obtenidos . . . . .	61
4.6. Validación del enclavamiento . . . . .	62
4.6.1. Testbench del módulo enclavamiento . . . . .	62
4.6.2. Resultados obtenidos . . . . .	66
4.7. Validación del registro . . . . .	67
4.7.1. Testbench del módulo registro . . . . .	67
4.7.2. Resultados obtenidos . . . . .	69
4.8. Validación del selector . . . . .	69
4.8.1. Testbench del módulo selector . . . . .	70
4.8.2. Resultados obtenidos . . . . .	72
<b>5. Conclusiones</b>	75
5.1. Resultados obtenidos . . . . .	75
5.2. Próximos pasos . . . . .	76
<b>Bibliografía</b>	77

## Índice de figuras

1.1. Ejemplo de implementación de topología sencilla . . . . .	3
1.2. <b>Ejemplo de implementación de topología compleja.</b> . . . . .	3
1.3. Proceso de generación automática de la solución electrónica . . . . .	4
1.4. Tramo de vía ferroviaria . . . . .	4
1.5. (a) Semáforo de tres aspectos (b) Semáforo doble de tres aspectos (Estación Olivos). . . . .	5
1.6. (a) Semáforo de dos aspectos (b) Semáforos de cruce de vías (Estación Lavallol). . . . .	6
1.7. Ocupación de las secciones de vías . . . . .	6
1.8. Estado de los aspectos ferroviarios según la ubicación del tren . . . . .	7
1.9. Pasos a nivel vehicular y peatonal sobre vía férrea . . . . .	8
1.10. Máquina de cambios de Lavallol (Línea Roca) . . . . .	8
1.11. Cambio de vías de estación Matheu (Línea Mitre) . . . . .	9
1.12. Posiciones normal e inversa del cambio . . . . .	9
1.13. Vía simple con <i>bypass</i> . . . . .	10
1.14. Sistema mecánico de enclavamiento en la antigua estación de Chascomús, hoy convertida en museo . . . . .	11
1.15. Bastidor de relés de estación Lavallol (Línea Roca) . . . . .	11
1.16. Panel de control enclavamientos - Central Lavallol . . . . .	12
1.17. Redundancia por votación 2 de 3 . . . . .	12
2.1. Topología bypass . . . . .	16
2.2. Topología estación con andén único . . . . .	16
2.3. Topología hub . . . . .	17
2.4. Topología terminal . . . . .	17
2.5. Ejemplo de elaboración de las rutas . . . . .	18
2.6. Enfoque funcional . . . . .	20
2.7. Esquema de trabajo en el enfoque funcional . . . . .	20
2.8. Escalabilidad del enfoque funcional . . . . .	21
2.9. Enfoque geográfico . . . . .	22
2.10. Pasaje de topología ferroviaria a grafo . . . . .	23
2.11. Análisis del grafo ferroviario . . . . .	23
2.12. Esquema de trabajo en el enfoque geográfico . . . . .	24
2.13. Escalabilidad del enfoque geográfico . . . . .	25
3.1. Diagrama en bloques genérico de una FSMD . . . . .	27
3.2. FPGA Arty Z7-10 de Digilent . . . . .	28
3.3. Grafo luego de ser analizado por el algoritmo . . . . .	29
3.4. FSMD del módulo de nodo genérico . . . . .	30
3.5. Conexión del módulo de la máquina de cambios . . . . .	31
3.6. FSMD del módulo de máquina de cambios . . . . .	31
3.7. FSMD del módulo separador . . . . .	32
3.8. FSMD del módulo mediador . . . . .	33

3.9. FSMD del módulo detector . . . . .	34
3.10. Estados del módulo detector . . . . .	34
3.11. FSMD del módulo registro . . . . .	35
3.12. Estados del módulo registro . . . . .	36
3.13. Diagrama de estados finitos digitales del módulo selector . . . . .	37
3.14. Diagrama en bloques de la UART . . . . .	38
3.15. Menú de opciones para comunicarse con el sistema . . . . .	39
4.1. Simulación de un nodo . . . . .	45
4.2. Simulación de la máquina de cambios . . . . .	49
4.3. Simulación de la UART . . . . .	55
4.4. Simulación del detector . . . . .	59
4.5. Simulación del separador . . . . .	64
4.6. Simulación del enclavamiento . . . . .	65
4.7. Simulación del mediador . . . . .	65
4.8. Simulación del registro . . . . .	67
4.9. Simulación del selector . . . . .	71

## Índice de figuras

1.1. Ejemplo de implementación de Belgrano R . . . . .	3
1.2. Proceso de generación automática de la solución electrónica . . . . .	3
1.3. Analizador de redes ferroviarias (Python) . . . . .	4
1.4. Archivos VHDL generados por el analizador de redes ferroviarias . . . . .	4
1.5. Imágenes generadas para todas las topologías . . . . .	5
1.6. Esquemático generado por el analizador de redes ferroviarias . . . . .	5
1.7. Tramo de vía ferroviaria . . . . .	6
1.8. (a) Semáforo de tres aspectos (b) Semáforo doble de tres aspectos (Estación Olivos) . . . . .	7
1.9. (a) Semáforos de dos aspectos (b) Semáforos de cruce de vías (Estación Lavallol) . . . . .	7
1.10. Ocupación de las secciones de vías . . . . .	8
1.11. Estado de los aspectos ferroviarios según la ubicación del tren . . . . .	8
1.12. Pasos a nivel vehicular y peatonal sobre vía férrea . . . . .	9
1.13. Máquina de cambios de Lavallol (Línea Roca) . . . . .	10
1.14. Cambio de vías de estación Matheu (Línea Mitre) . . . . .	10
1.15. Posiciones normal e inversa del cambio . . . . .	11
1.16. Vía simple con <i>bypass</i> . . . . .	11
1.17. Sistema enclavamiento mecánico en la estación de Chascomús, hoy convertida en museo . . . . .	12
1.18. Bastidor de relés de estación Lavallol (Línea Roca) . . . . .	13
1.19. Panel de control enclavamientos - Central Lavallol . . . . .	13
1.20. Redundancia por votación 2 de 3 . . . . .	14
2.1. Topología bypass . . . . .	16
2.2. Topología estación con andén único . . . . .	16
2.3. Topología hub . . . . .	17
2.4. Topología terminal . . . . .	17
2.5. Ejemplo de elaboración de las rutas . . . . .	19
2.6. Enfoque funcional . . . . .	20
2.7. Esquema de trabajo en el enfoque funcional . . . . .	21
2.8. Escalabilidad del enfoque funcional . . . . .	22
2.9. Enfoque geográfico . . . . .	23
2.10. Pasaje de topología ferroviaria a grafo . . . . .	23
2.11. Análisis del grafo ferroviario . . . . .	24
2.12. Esquema de trabajo en el enfoque geográfico . . . . .	25
2.13. Escalabilidad del enfoque geográfico . . . . .	25
2.14. Diagrama en bloques genérico de una FSMD . . . . .	26
2.15. FPGA Arty Z7-10 de Digilent . . . . .	27
3.1. Grafo luego de ser analizado por el algoritmo . . . . .	30
3.2. FSMD del módulo de nodo genérico . . . . .	32
3.3. Conexión del módulo de la máquina de cambios . . . . .	32

3.4. FSMD del módulo de máquina de cambios . . . . .	33
3.5. FSMD del módulo separador . . . . .	35
3.6. FSMD del módulo mediador . . . . .	35
3.7. FSMD del módulo detector . . . . .	36
3.8. Estados del módulo detector . . . . .	37
3.9. FSMD del módulo registro . . . . .	38
3.10. Estados del módulo registro . . . . .	39
3.11. Diagrama de estados finitos digitales del módulo selector . . . . .	40
3.12. Diagrama en bloques de la UART . . . . .	41
3.13. Menú de opciones para comunicarse con el sistema . . . . .	42
3.14. Grafo modificado en tiempo real por la FPGA . . . . .	42
4.1. Simulación de un nodo . . . . .	47
4.2. Simulación de la máquina de cambios . . . . .	51
4.3. Simulación de la UART . . . . .	57
4.4. Simulación del detector . . . . .	61
4.5. Simulación del separador . . . . .	66
4.6. Simulación del enclavamiento . . . . .	67
4.7. Simulación del mediador . . . . .	67
4.8. Simulación del registro . . . . .	69
4.9. Simulación del selector . . . . .	73

## Índice de Tablas

2.1. Tabla de enclavamientos (caso unidireccional) . . . . .	18
2.2. Tabla de enclavamientos (caso bidireccional) . . . . .	19
3.1. Trama de <b>datos</b> a recibir . . . . .	32
3.2. Trama de <b>datos</b> a enviar . . . . .	32
4.1. Combinaciones posibles . . . . .	42
4.2. Combinaciones posibles . . . . .	45

## Índice de Tablas

2.1. Tabla de enclavamientos (izquierda a derecha) . . . . .	19
2.2. Tabla de enclavamientos (derecha a izquierda) . . . . .	19
2.3. Tabla de enclavamientos (caso bidireccional) . . . . .	20
3.1. Convención para elementos ferroviarios I . . . . .	33
3.2. Convención para elementos ferroviarios II . . . . .	34
3.3. Trama de entrada [N] . . . . .	34
3.4. Trama de salida [M] . . . . .	34
4.1. Combinaciones posibles . . . . .	44
4.2. Combinaciones posibles . . . . .	47

XIII

*"Él hará la unidad de la República mejor que todos los Congresos. Estos podrán declararla una e indivisible pero sin el camino de fierro que acerque sus extremos remotos, quedará siempre divisible y dividida contra todos los Decretos Legislativos."*

Juan Bautista Alberdi (1810-1884).

XIII

*"Él hará la unidad de la República mejor que todos los Congresos. Estos podrán declararla una e indivisible pero sin el camino de fierro que acerque sus extremos remotos, quedará siempre divisible y dividida contra todos los Decretos Legislativos."*

Juan Bautista Alberdi (1810-1884).

## Capítulo 1

### Introducción General

En este capítulo se presentan el contexto del proyecto, los sistemas ferroviarios, una introducción a las distintas tecnologías de los sistemas de enclavamientos y los objetivos a cumplir.

#### 1.1. Contexto y motivación

En la actualidad, el sistema ferroviario argentino se encuentra enormemente deteriorado y desactualizado. Mientras que otras naciones más avanzadas han progresado en la migración de sus sistemas de transporte de cargas y pasajeros añadiendo electrónica de última generación, Argentina continúa utilizando mecanismos diseñados a principios o mediados del siglo XX.

Esto repercute negativamente en la poca o nula seguridad que dichos sistemas pueden brindar a los pasajeros, conductores, peatones y automovilistas. Pero los sistemas electrónicos para la seguridad vial de trenes y subtes son importados y muy costosos, además de estar monopolarizados por menos de una docena de empresas en todo el mundo. Un sistema de enclavamiento como el que se aborda en este trabajo puede costar entre 5 y 10 millones de dólares[1] y se requieren varias decenas de estos sistemas sólo para la zona urbana de la Ciudad Autónoma de Buenos Aires.

Dichas empresas brindan muy poca información técnica sobre sus desarrollos y la mayoría de las herramientas que utilizan son de uso privado empresarial. No obstante, deben regirse bajo normativas como las que la Comunidad Europea ha establecido desde 2004. Las tres normas principales son EN-50126[2] (ciclo de vida), EN-50128[3] (técnicas de software) y EN-50129[4] (técnicas de hardware). Entre otras cuestiones, se busca la integridad de la seguridad.

En los sistemas críticos, una falla puede poner en peligro cientos de vidas humanas y/o costosas inversiones. Por lo tanto, se deben cumplir estrictos parámetros de fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS, del inglés *Reliability, Availability, Maintainability and Safety*), durante todo el ciclo de vida.

En este contexto, en 2015 se creó el CONICET-GICSAFe, cuyas siglas corresponden a Grupo de Investigación en Calidad y Seguridad de las Aplicaciones Ferroviarias, conformado por docentes e investigadores de una docena de universidades e instituciones públicas argentinas[5]. El grupo desarrolla sistemas electrónicos e informáticos para aplicaciones ferroviarias relacionadas con la seguridad, a partir de la generación de un prototipo funcional y la documentación correspondiente que luego se transfiere en su totalidad a los clientes. En particular, esta metodología se utiliza en este trabajo con Trenes Argentinos, que es la Sociedad

## Capítulo 1

### Introducción General

En este capítulo se presentan el contexto del proyecto, los sistemas ferroviarios, las distintas tecnologías de los sistemas de enclavamientos y los objetivos a cumplir.

#### 1.1. Contexto y motivación

Mientras que otras naciones han progresado en la migración de sus sistemas de transporte de cargas y pasajeros añadiendo electrónica de última generación, Argentina en muchos casos continúa utilizando mecanismos diseñados a principios o mediados del siglo XX[1, 2]. Esto repercute negativamente en la seguridad que dichos sistemas pueden brindar a los pasajeros, conductores, peatones y automovilistas. A la vez, ocurre que los sistemas electrónicos para la seguridad vial de trenes y subtes son importados y muy costosos, además de constituir un oligopolio conformado por menos de una docena de empresas en todo el mundo. Un sistema de enclavamiento como el que se aborda en este trabajo puede costar entre 5 y 10 millones de dólares[3] y se requieren varias decenas de estos sistemas sólo para la zona urbana de la Ciudad Autónoma de Buenos Aires.

Las empresas que fabrican sistemas de enclavamiento [4-13] brindan muy poca información técnica sobre sus desarrollos y la mayoría de las herramientas que utilizan son de uso privado de cada empresa. No obstante, deben regirse bajo normativas como las que la Unión Europea ha establecido desde 2004. Las tres normas principales son EN-50126[4] (ciclo de vida), EN-50128[5] (técnicas de software) y EN-50129[6] (técnicas de hardware).

Los sistemas de enclavamiento son sistemas críticos. Esto implica que una falla puede poner en peligro cientos de vidas humanas y/o costosas inversiones. Por lo tanto, se deben cumplir estrictos parámetros de fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS, del inglés *Reliability, Availability, Maintainability and Safety*), durante todo el ciclo de vida.

En este contexto, en 2015 se creó el CONICET-GICSAFe [17], cuyas siglas corresponden a Grupo de Investigación en Calidad y Seguridad de las Aplicaciones Ferroviarias, conformado por docentes e investigadores de una docena de universidades e instituciones públicas argentinas. El grupo desarrolla sistemas electrónicos e informáticos para aplicaciones ferroviarias relacionadas con la seguridad, a partir de la generación de un prototipo funcional y la documentación correspondiente que luego se transfiere en su totalidad a los clientes. En particular, esta metodología se utiliza en este trabajo con Trenes Argentinos[18], que es la Sociedad del Estado que opera las líneas Roca, Sarmiento, Mitre, San Martín y

del Estado que opera las líneas Roca, Sarmiento, Mitre, San Martín y Belgrano Sur, entre otras. Luego el cliente puede fabricar el sistema diseñado por el GICSAFe o solicitar su fabricación, así como modificar el sistema de acuerdo con sus necesidades.

En el marco de la Especialización de Sistemas Embebidos, desde julio de 2018 se tuvieron reuniones con diferentes funcionarios y profesionales de Trenes Argentinos. En particular los encuentros se desarrollaron con la Gerencia de Ingeniería, Gerencia de Seguridad Operacional, Subgerencia de Desarrollo y Normas Técnicas, Subgerencia de Transporte, Gerencia de Señalamiento, entre otros, de los cuales surgió el interés en el desarrollo del presente proyecto. De dichas visitas y otras posteriores se obtuvo la totalidad de las fotografías incluidas en esta memoria.

El desarrollo del sistema involucra muchas áreas distintas: procesamiento, comunicación, replicación del sistema para otras topologías, testing, etc. Un primer prototípico se desarrolló a finales de 2018 y actualmente se culminó una segunda versión para la Maestría en Sistemas Embebidos. La amplia variedad de topologías existentes obligó a abandonar la metodología de desarrollo *artesanal* de cada sistema y fue necesario abordar el diseño de forma general, para topologías genéricas, lo que adiciona aún más trabajo al monumental proyecto.

Por esa razón, durante la Maestría en Sistemas Embebidos, en el año 2019, se comenzó a seguir un estrategia orientada a obtener una solución integral para cualquier ubicación, generando automáticamente el código y los testbenches involucrados. A la vez que miembros de CONICET-GICSAFe pertenecientes a UTN-Haedo comenzaron el desarrollo de un front-end gráfico y su correspondiente simulador en tiempo real, en vistas a ser integrado al proyecto en el mediano plazo.

Hacia finales de 2019 se iniciaron reuniones con miembros de la Comisión Nacional de Energía Atómica (CNEA), para integrar el proyecto en sus plataformas de hardware ampliamente testeadas en el ámbito de los sistemas críticos. Además del intercambio de conocimiento y la puesta en común de estrategias a utilizar, se aprovechó todo lo posible la amplia experiencia que ellos brindaron a GICSAFe

Belgrano Sur, entre otras. Luego el cliente puede fabricar el sistema diseñado por el GICSAFe o solicitar su fabricación, así como modificar el sistema de acuerdo con sus necesidades.

En el marco de la Especialización de Sistemas Embebidos, desde julio de 2018 se tuvieron reuniones con diferentes funcionarios y profesionales de Trenes Argentinos. En particular los encuentros se desarrollaron con la Gerencia de Ingeniería, Gerencia de Seguridad Operacional, Subgerencia de Desarrollo y Normas Técnicas, Subgerencia de Transporte, Gerencia de Señalamiento, entre otros, de los cuales surgió el interés en el desarrollo del presente proyecto. De dichas visitas y otras posteriores se obtuvo la totalidad de las fotografías incluidas en esta memoria.

El desarrollo del sistema involucra muchas áreas distintas: procesamiento, comunicación, replicación del sistema para otras topologías, testing, etc. Un primer prototípico se desarrolló a finales de 2018 y actualmente se culminó una segunda versión para la Maestría en Sistemas Embebidos. La amplia variedad de topologías existentes obligó a abandonar la metodología de desarrollo *ad hoc* de cada sistema y fue necesario abordar el diseño de forma general, para topologías genéricas, lo que adiciona aún más trabajo al proyecto.

Por esa razón, durante la Maestría en Sistemas Embebidos, en el año 2019, se comenzó a seguir un estrategia orientada a obtener una solución integral para cualquier ubicación, generando automáticamente el código y los testbenches involucrados. A la vez que miembros de CONICET-GICSAFe pertenecientes a UTN-Haedo comenzaron el desarrollo de un front-end gráfico y su correspondiente simulador en tiempo real, en vistas a ser integrado al proyecto en el mediano plazo.

Hacia finales de 2019 se iniciaron reuniones con miembros de la Comisión Nacional de Energía Atómica (CNEA), para integrar el proyecto en sus plataformas de hardware ampliamente testeadas en el ámbito de los sistemas críticos. Además del intercambio de conocimiento y la puesta en común de estrategias a utilizar, se aprovechó todo lo posible la amplia experiencia que ellos brindaron a GICSAFe

durante el pasado y corriente año.

## 1.2. Propuesta de solución

El proyecto comenzó en 2018 en el marco de la Especialización de Sistemas Embebidos. Durante ese año se implementó un sistema similar al de la Figura 1.1. El sistema contaba con una cantidad limitada de tramos de vías y bifurcaciones que implicaba una cantidad acotada de circuitos lógicos complicados pero posibles de obtener y ensayar.

La implementación se hacía de forma artesanal, bloque a bloque y conectándolos de igual manera uno por uno. Diseñar los ensayos demoraba días o semanas y era necesario revisar varias veces las pocas y difusas especificaciones que se tenían para corroborar un correcto funcionamiento del sistema.

### 1.2. Propuesta de solución

3

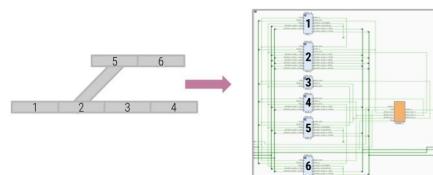


FIGURA 1.1: Ejemplo de implementación de topología sencilla.

Durante la especialización fue una constante que la locación en la cual se basaba el desarrollo del sistema cambiaba abruptamente por otra diferente, representada en la figura 1.2. Por lo que, aun manteniendo los mismos conceptos iniciales, el alcance y los requerimientos cambiaban completamente y se tenía que desechar gran parte de lo construido, tanto del sistema como de los tests, que ya no tenían sentido para el nuevo sistema. El rediseñar de todo de cero fue algo común durante esta etapa del proyecto.

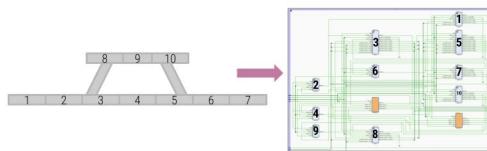


FIGURA 1.2: Ejemplo de implementación de topología compleja.

Una mayor cantidad de elementos ferroviarios implica una densidad mayor de circuitos lógicos, y estos a su vez repercuten en un crecimiento exponencial de la complejidad del desarrollo, mayor dificultad para implementar los ensayos y mayores chances de errores al extrapolarse conceptos aún inmaduros a sistemas de mayor tamaño.

La solución a proponer no podía quedar sujetada a una única topología, ya que aun culminando el proyecto se corría el riesgo de que el tiempo empleado fuese desperdiciado si se cambiaban los requerimientos de forma tan abrupta. Es por eso que se añadió como objetivo de este trabajo el desarrollo de una herramienta capaz de generar automáticamente la solución electrónica de un sistema de enclavamiento ferroviario, a partir de una representación matemática única de cada topología. En la figura 1.3 se puede visualizar el proceso.

De esta manera, la solución parte de cualquier red ferroviaria, representada mediante un grafo y un analizador de redes ferroviarias (desarrollado también en este trabajo), detecta qué función cumplirá cada elemento de la red y determina cuántos semáforos, de cuántos aspectos, en qué orientación y en qué nodo deben

4

## Capítulo 1. Introducción General

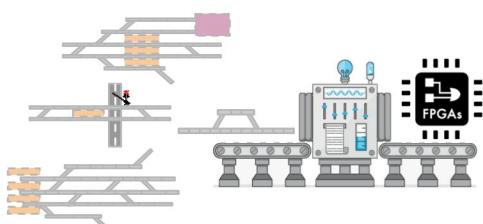


FIGURA 1.3: Proceso de generación automática de la solución electrónica.

situarse para que el sistema sea seguro. A continuación, en base a los semáforos insertados el analizador calcula todas las rutas posibles que admite esa red.

Para poder entender mejor el funcionamiento del sistema es necesario introducir conceptos propios del mundo ferroviario y sus componentes involucrados, que se describen en la sección siguiente.

## 1.3. Elementos del señalamiento ferroviario

La función del señalamiento ferroviario es evitar las colisiones entre trenes y los descarrilamientos. A continuación se describen los diferentes elementos que com-

des de fines del año pasado a la recta.

## 1.2. Propuesta de solución

En el marco de la Especialización de Sistemas Embebidos se implementó en 2018 un sistema de enclavamiento para la estación Belgrano R de la Línea Mitre, como puede verse en la Figura 1.1. El mismo contaba con una cantidad limitada de tramos de vías y bifurcaciones que implicaba una cantidad acotada de circuitos lógicos.

La implementación se hacía de forma ad hoc, bloque a bloque y conectándolos de igual manera uno por uno. Diseñar los ensayos demoraba días o semanas y era necesario revisar varias veces las pocas y difusas especificaciones que se tenían para corroborar un correcto funcionamiento del sistema.

El desarrollo a medida de la estación Belgrano R permitió afianzar muchos conceptos del mundo ferroviario sobre cómo se comportaban sus diversos componentes y como interactuaban a nivel general. No obstante, de querer replicar el

### 1.2. Propuesta de solución

3

desarrollo en otras locaciones, aún con mínimas diferencias con Belgrano R, hubiese implicado rehacer por completo todo el proceso: análisis, diseño, testing, etc.

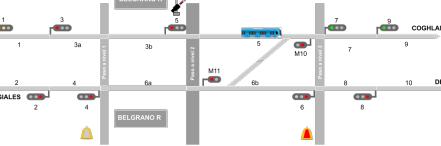


FIGURA 1.1: Ejemplo de implementación de Belgrano R.

Fue necesario garantizar no solo la reusabilidad, en el caso de locaciones de un tamaño similar a Belgrano R; sino que también su escalabilidad a topologías más complejas. Una mayor cantidad de elementos ferroviarios implica una densidad mayor de circuitos lógicos, y estos a su vez repercuten en un crecimiento enormemente de la complejidad del desarrollo, mayor dificultad para implementar los ensayos y mayores chances de errores al extrapolarse posibles fallas humanas a sistemas de mayor tamaño.

La solución a proponer no podía quedar sujetada a una única topología, ya que aun culminando el proyecto se corría el riesgo de que el tiempo empleado fuese desperdiciado si se cambiaban los requerimientos de forma abrupta. Es por eso que, en el marco de la Maestría en Sistemas Embebidos, se marcó como un objetivo el desarrollo de una herramienta capaz de generar automáticamente la solución electrónica de un sistema de enclavamiento ferroviario, a partir de una representación matemática única de cada topología. En la figura 1.2 se puede visualizar el proceso, en el cual diversas topologías son procesadas e implementadas en una plataforma FPGA.

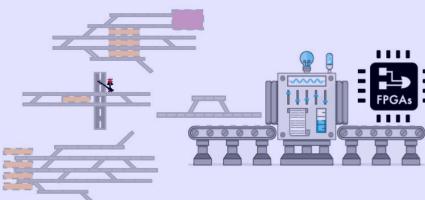
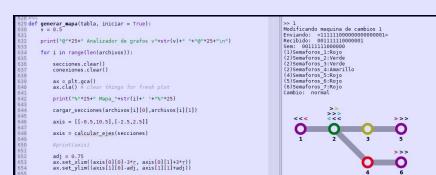


FIGURA 1.2: Proceso de generación automática de la solución electrónica.

## 4 Capítulo 1. Introducción General

De esta manera, la solución parte de cualquier red ferroviaria, representada mediante un grafo y un analizador de redes ferroviarias (figura 1.3), detecta qué función cumplirá cada elemento de la red y determina cuántos semáforos, de cuántos aspectos, en qué orientación y en qué nodo deben situarse para que el sistema sea seguro.



ponente el señalamiento y que fueron modelados durante este trabajo.

### 1.3.1. Vías

Las vías férreas (figura 1.4) consisten en el elemento esencial de la infraestructura ferroviaria y conforman el sitio por el cual se desplazan los trenes. Se encuentran separadas por una distancia fija que se mide desde sus caras internas y se denomina trocha.



FIGURA 1.4: Tramo de vía ferroviaria.

```
    ajuste = 0
    ajuste = 3
    bajar_revolucion()
    subir_revolucion()
    codice_revolucion()
    codice_revolucion()
    subir_revolucion()
    motor_revoluciones(1,gif == False)
    motor_revoluciones(1,gif == False)
    detectar_girofaro(cambiar)
    print("a = ", a, " b = ", b, " c = ", [a-0.03, b-0.02, c-0.03])
```

FIGURA 1.3: Analizador de redes ferroviarias (Python).

A continuación, en base a los semáforos insertados al analizador calcula todas las rutas posibles que admite esa red y genera automáticamente la solución electrónica para ser implementada mediante una FPGA. Puede verse en la figura 1.4 que la cantidad de archivos creados es considerable, pero todo es producto del análisis de la red y son todos los códigos necesarios para implementar el sistema.



**FIGURA 1.4:** Archivos VHDL generados por el analizador de redes ferroviarias.

El analizador de redes ferroviarias no solo procesa una topología, sino todas las

### **1.2. Propuesta de solución**

5

definidas en archivos individuales. Es decir, si la carpeta principal tiene  $N$  archivos de texto plano que describen  $N$  topologías, el analizador generará  $N$  análisis distintos y por cada uno presentará una imagen de la topología recibida, como puede verse en la figura 1.5. De la misma manera genera un historial técnico del funcionamiento del sistema llamado tabla de enclavamientos [19, 20].

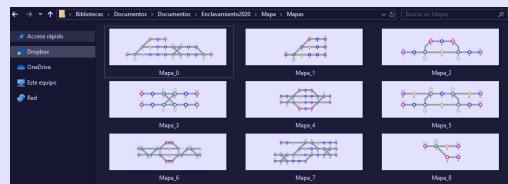


FIGURA 1.5: Imágenes generadas para todas las topologías.

Finalmente, todo el proceso culmina con la creación del sistema electrónico, del cual se presenta un esquema de ejemplo en la figura 1.6

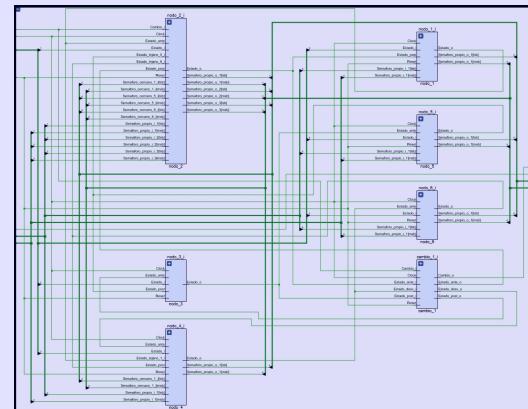


FIGURA 1.6: Esquemático generado por el analizador de redes ferroviarias

Para poder entender mejor el funcionamiento del sistema es necesario introducir conceptos propios del mundo ferroviario y sus componentes involucrados, que se describen en la sección siguiente.

### 1.3. Elementos del señalamiento ferroviario

En este trabajo se reutilizan las mismas definiciones de los elementos ferroviarios presentadas en la memoria del Trabajo Final de la Carrera de Especialización en Sistemas Embobidos, ya que son las definiciones mas precisas que este autor pudo elaborar y el paso del tiempo no ha modificado su validez.

La función del señalamiento ferroviario es evitar las colisiones entre trenes y los descarrilamientos. A continuación se describen los diferentes elementos que componen el señalamiento y que fueron modelados durante este trabajo.

### 1.3.1. Vía

Las vías férreas (figura 1.7) consisten en el elemento esencial de la infraestructura

### 1.3. Elementos del señalamiento ferroviario

5

Las vías se dividen en secciones y por seguridad se establece que cada sección puede contener solo una formación por vez. Las mismas pueden tener largos variables en zonas urbanas de entre 500 a 1000 metros en zonas rurales.

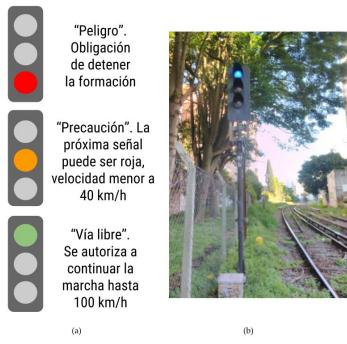
Cada vía puede ser clasificada en dos grupos: vías ascendentes o vías descendentes. Las ascendentes son aquellas por las que los trenes circulan únicamente en la dirección del kilometraje en sentido creciente. Las descendentes son aquellas por las que los trenes circulan únicamente en la dirección del kilometraje en sentido decreciente[6]. El kilómetro 0 es la estación principal de la línea ferroviaria, como por ejemplo: Plaza Constitución (para la línea Roca), Once de septiembre (para la línea Sarmiento) o Retiro (para las líneas Mitre y San Martín).

Existen vías de maniobra que pueden ser tanto ascendentes como descendentes. Estas vinculan, mediante un cambio de vías, una sección ascendente con otra descendente, en la cual los trenes deben circular a una velocidad reducida.

#### 1.3.2. Semáforos ferroviarios

El sistema de enclavamientos utiliza los semáforos ferroviarios para indicarle al conductor del tren si puede o no acceder al próximo tramo de vías y a qué velocidad se le permite circular; esto, por medio del color del semáforo, denominado aspecto. A diferencia de los semáforos vehiculares, en los que cada color es alternado por otro de la secuencia rojo-amarillo-verde en función del tiempo, los semáforos ferroviarios cambian su aspecto en función de los eventos de los tramos siguientes.

En la figura 1.5 se presenta un esquema de señales de tres aspectos, que es el tipo de semáforo que se utiliza en la gran mayoría de las líneas ferroviarias.



6

### Capítulo 1. Introducción General

Otra diferencia fundamental es que no todos los semáforos ferroviarios poseen tres aspectos. Los semáforos de maniobras constan de solo dos, amarillo (precaución) y rojo (prohibido avanzar), y algunas líneas como la Línea Roca utilizan semáforos de cuatro aspectos.

En la figura 1.6 se visualizan los semáforos de dos aspectos. Se utilizan en cambios de vías donde, por su peligrosidad, solo se podrían permitir aspectos rojos y amarillos.



Los semáforos de cuatro aspectos son utilizados en la Línea Roca y poseen un doble amarillo antes del amarillo simple, para permitir así tramos de vías más cortos en forma segura. Como no son objeto de estudio del presente trabajo, no serán explicados aquí.

#### 1.3.3. Circuito de vías

Para poder determinar si un tramo de la vía se encuentra ocupado o libre se utilizan los circuitos de vías. Estos constituyen componentes electrónicos que impiden una tensión conocida entre los rieles, y cuando un tren se posiciona sobre esa sección provoca un cortocircuito que es detectado por el circuito. En la figura 1.7 se ejemplifica la ocupación de las secciones por una formación (modelada con un 0) y la ausencia de formación (modelada con un 1).



ferroviaria y conforman el sitio por el cual se desplazan los trenes. Se encuentran separadas por una distancia fija que se mide desde sus caras internas y se denomina trocha.



FIGURA 1.7: Tramo de vía ferroviaria.

Las vías se dividen en secciones y por seguridad se establece que cada sección puede contener solo una formación por vez. Las mismas pueden tener largos variables en zonas urbanas de entre 500 a 1000 metros en zonas rurales.

Cada vía puede ser clasificada en dos grupos: vías ascendentes o vías descendentes. Las ascendentes son aquellas por las que los trenes circulan únicamente en la dirección del kilometraje en sentido creciente. Las descendentes son aquellas por las que los trenes circulan únicamente en la dirección del kilometraje en sentido decreciente[21]. El kilómetro 0 es la estación principal de la línea ferroviaria, como por ejemplo: Plaza Constitución (para la línea Roca), Once de septiembre (para la línea Sarmiento) o Retiro (para las líneas Mitre y San Martín).

Existen vías de maniobra que pueden ser tanto ascendentes como descendentes. Estas vinculan, mediante un cambio de vías, una sección ascendente con otra descendente, en la cual los trenes deben circular a una velocidad reducida.

#### 1.3.2. Semáforos ferroviarios

El sistema de enclavamientos utiliza los semáforos ferroviarios para indicarle al conductor del tren si puede o no acceder al próximo tramo de vías y a qué velocidad se le permite circular; esto, por medio del color del semáforo, denominado aspecto

### 1.3. Elementos del señalamiento ferroviario

7

aspecto. A diferencia de los semáforos vehiculares, en los que cada color es alternado por otro de la secuencia rojo-amarillo-verde en función del tiempo, los semáforos ferroviarios cambian su aspecto en función de los eventos de los tramos siguientes.

En la figura 1.8 se presenta un esquema de señales de tres aspectos, que es el tipo de semáforo que se utiliza en la gran mayoría de las líneas ferroviarias.



FIGURA 1.8: (a) Semáforo de tres aspectos  
(b) Semáforo doble de tres aspectos (Estación Olivos).

Otra diferencia fundamental es que no todos los semáforos ferroviarios poseen tres aspectos. Los semáforos de maniobras constan de solo dos, amarillo (precaución) y rojo (prohibido avanzar), y algunas líneas como la Línea Roca utilizan semáforos de cuatro aspectos.

En la figura 1.9 se visualizan los semáforos de dos aspectos. Se utilizan en cambios de vías donde, por su peligrosidad, solo se podrían permitir aspectos rojos y amarillos.

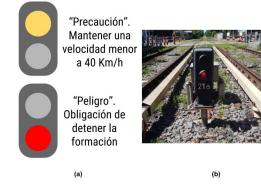


FIGURA 1.9: (a) Semáforo de dos aspectos  
(b) Semáforos de cruce de vías (Estación Lavallol).

Los semáforos de cuatro aspectos son utilizados en la Línea Roca y poseen un doble amarillo antes del amarillo simple, para permitir así tramos de vías más cortos en forma segura. Como no son objeto de estudio del presente trabajo, no serán explicados aquí.

#### 1.3.3. Circuito de vías

Para poder determinar si un tramo de la vía se encuentra ocupado o libre se utilizan los circuitos de vías. Estos constituyen componentes electrónicos que impiden una tensión conocida entre los rieles, y cuando un tren se posiciona sobre esa sección provoca un cortocircuito que es detectado por el circuito. En la figura 1.10 se ejemplifica la ocupación de las secciones por una formación (modelada con un 0) y la ausencia de formación (modelada con un 1).

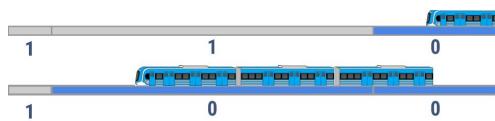


FIGURA 1.7: Ocupación de las secciones de vías.

## 1.3. Elementos del señalamiento ferroviario

7

Si el tramo de vía no tiene ninguna formación ocupándolo, el señalamiento indicará un aspecto verde o amarillo según el estado de ocupación del tramo siguiente. Si la formación ocupa la sección, el señalamiento cambiará su aspecto a rojo para indicar que no puede ingresar ninguna otra formación, a fin de evitar colisiones. Por seguridad también se establecerá a rojo el semáforo anterior y a amarillo el anterior a este (doble recubrimiento), tal como se ilustra en la figura 1.8.

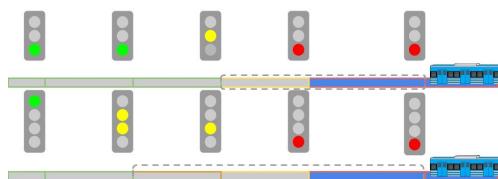


FIGURA 1.8: Estado de los aspectos ferroviarios según la ubicación del tren.

Si la alimentación es interrumpida o si el cableado sufre alguna falla, entonces el sistema asumirá que hay un tren en las vías y los semáforos se pondrán en aspecto rojo para que las formaciones cercanas detengan su marcha y las barreras de los pasos a nivel desciendan. A este principio se lo denomina *fail-safe*<sup>1</sup>. Es decir, si por alguna razón algo falla, el sistema adopta la condición más restrictiva, mitigando la posibilidad de una situación peligrosa.

## 1.3.4. Pasos a nivel

La intersección de una ruta vehicular o peatonal con la vía férrea se denomina paso a nivel. El sistema de control de la barrera mantiene el brazo de esta en alto para permitir la circulación vehicular, como se puede ver en la figura 1.9. Si un tren ocupa las secciones amarillas de la figura 1.9 se desenergiza la barrera y comienza a descender el brazo por efecto de la gravedad. Cuando se ocupen las secciones azules de la figura 1.9, entonces se accionará la alarma sonora para alertar a los peatones que deben permanecer en el laberinto contiguo a la vía, cuya función es forzar a los peatones a mirar a ambos lados antes de cruzar el paso a nivel.

Solo cuando la barrera baja, el tren tiene permitido avanzar sobre el cruce, siendo el paso a nivel un sector de altísimo riesgo.

Al desocuparse las secciones amarillas, la barrera vuelve a energizarse y se sitúa en estado alto nuevamente, a la espera de otro tren para reiniciar el proceso descripto.

Se debe destacar que el mismo proceso de descenso de la barrera ocurrirá si esta se desenergiza por una falla eléctrica y/o pérdida de alimentación. Es decir, el

<sup>1</sup>*fail-safe*: falla segura.

8

## Capítulo 1. Introducción General

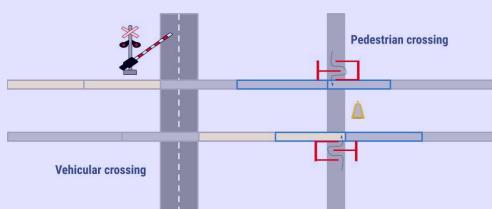


FIGURA 1.9: Pasos a nivel vehicular y peatonal sobre vía férrea.

sistema asumirá el estado más seguro ante cualquiera de los mencionados fallos, siguiendo el principio de falla segura.

## 1.3.5. Máquina de cambios

Una máquina de cambios (figura 1.10) es un mecanismo utilizado para permitir el paso de las formaciones de una vía a una ramificación del recorrido principal. Esto se realiza mediante el movimiento de la aguja del cambio (riel móvil) hacia su respectiva contraaguja (riel fijo) hasta obtener un adecuado acoplamiento que permita la circulación de la formación.



se ejemplifica la ocupación de las secciones por una formación (modelada con un 0) y la ausencia de formación (modelada con un 1).

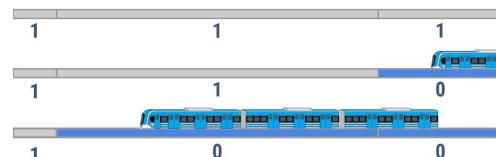


FIGURA 1.10: Ocupación de las secciones de vías.

Si el tramo de vía no tiene ninguna formación ocupándolo, el señalamiento indicará un aspecto verde o amarillo según el estado de ocupación del tramo siguiente. Si la formación ocupa la sección, el señalamiento cambiará su aspecto a rojo para indicar que no puede ingresar ninguna otra formación, a fin de evitar colisiones. Por seguridad también se establecerá a rojo el semáforo anterior y a amarillo el anterior a este (doble recubrimiento), tal como se ilustra en la figura 1.11.

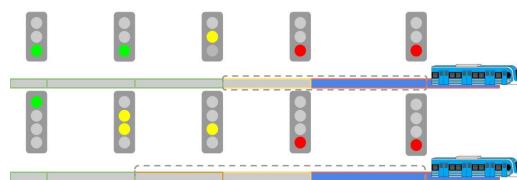


FIGURA 1.11: Estado de los aspectos ferroviarios según la ubicación del tren.

Si la alimentación es interrumpida o si el cableado sufre alguna falla, entonces el sistema asumirá que hay un tren en las vías y los semáforos se pondrán en aspecto rojo para que las formaciones cercanas detengan su marcha y las barreras de los

## 1.3. Elementos del señalamiento ferroviario

9

pasos a nivel desciendan. A este principio se lo denomina *fail-safe*<sup>1</sup>. Es decir, si por alguna razón algo falla, el sistema adopta la condición más restrictiva, mitigando la posibilidad de una situación peligrosa.

## 1.3.4. Pasos a nivel

La intersección de una ruta vehicular o peatonal con la vía férrea se denomina paso a nivel. El sistema de control de la barrera mantiene el brazo de esta en alto para permitir la circulación vehicular, como se puede ver en la figura 1.12. Si un tren ocupa las secciones amarillas de la figura 1.12 se desenergiza la barrera y comienza a descender el brazo por efecto de la gravedad. Cuando se ocupen las secciones azules de la figura 1.12, entonces se accionará la alarma sonora para alertar a los peatones que deben permanecer en el laberinto contiguo a la vía, cuya función es forzar a los peatones a mirar a ambos lados antes de cruzar el paso a nivel.

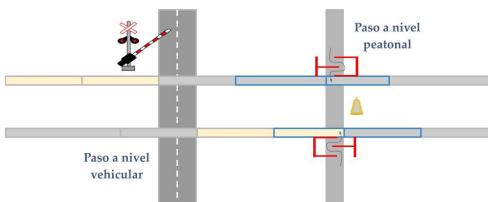


FIGURA 1.12: Pasos a nivel vehicular y peatonal sobre vía férrea.

Solo cuando la barrera baja, el tren tiene permitido avanzar sobre el cruce, siendo el paso a nivel un sector de altísimo riesgo.

Al desocuparse las secciones amarillas, la barrera vuelve a energizarse y se sitúa en estado alto nuevamente, a la espera de otro tren para reiniciar el proceso descripto.

Se debe destacar que el mismo proceso de descenso de la barrera ocurrirá si esta se desenergiza por una falla eléctrica y/o pérdida de alimentación. Es decir, el sistema asumirá el estado más seguro ante cualquiera de los mencionados fallos, siguiendo el principio de falla segura.

## 1.3.5. Máquina de cambios

Una máquina de cambios (figura 1.13) es un mecanismo utilizado para permitir el paso de las formaciones de una vía a una ramificación del recorrido principal. Esto se realiza mediante el movimiento de la aguja del cambio (riel móvil) hacia su respectiva contraaguja (riel fijo) hasta obtener un adecuado acoplamiento que permita la circulación de la formación.

<sup>1</sup>*fail-safe*: falla segura.

10

## Capítulo 1. Introducción General



FIGURA 1.10: Máquina de cambios de Lavallol (Línea Roca).



FIGURA 1.13: Máquina de cambios de Lavallol (Línea Roca).

En la figura 1.14 se muestra el cambio de vía de la estación Matheu de la Línea Mitre. Se observa que según sea la posición de la máquina de cambios, el tren puede continuar en la misma vía o hacer el cambio a la otra vía.



FIGURA 1.14: Cambio de vías de estación Matheu (Línea Mitre).

#### 1.4. Sistema de enclavamientos 9

En la figura 1.11 se muestra el cambio de vía de la estación Matheu de la Línea Mitre. Se observa que según sea la posición de la máquina de cambios, el tren puede continuar en la misma vía o hacer el cambio a la otra vía.



FIGURA 1.11: Cambio de vías de estación Matheu (Línea Mitre).

En la figura 1.12 se muestran las posiciones que puede adoptar el cambio. En la posición normal los trenes pueden circular de forma directa, en paralelo, por la vía principal en sentidos opuestos. En la posición reversa, en cambio, se permite el intercambio de trenes de una rama principal a otra en sentido opuesto o a una ramificación secundaria de la red.

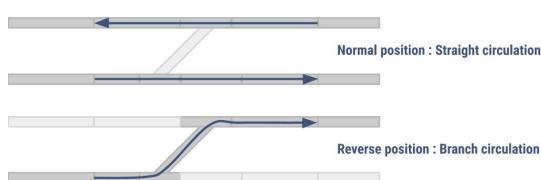


FIGURA 1.12: Posiciones normal e inversa del cambio.

#### 1.4. Sistema de enclavamientos

A modo de ejemplo se ilustra en la figura 2.11 un sistema de cambios en una vía simple con *bypass*. Este permite que dos formaciones puedan cruzarse en sentidos opuestos sin colisionar.

#### 10 Capítulo 1. Introducción General



FIGURA 1.13: Vía simple con bypass.

Para evitar la colisión, se requiere un control seguro que evite que las formaciones avancen hacia secciones ya ocupadas por otras. También debe evitar que las formaciones avancen sobre los cambios cuando estos aún no han terminado de posicionarse en su lugar, lo que provocaría descarrillamientos. A este control se lo denomina sistema de enclavamiento<sup>10</sup> impide que se produzcan las configuraciones no seguras y controla los semáforos que habilitan o no los itinerarios de las formaciones.

Una falla en un enclavamiento puede poner en peligro cientos de vidas humanas y generar gastos considerables. Por lo tanto, en el diseño del sistema de enclavamiento se deben cumplir estrictos parámetros de fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS).

Lamentablemente, los sistemas de enclavamientos en Argentina son en su mayoría mecánicos, de comienzos del siglo XX, y otra cantidad considerable son electromecánicos, de más de 40 años de antigüedad. Muchos de ellos ya han agotado su vida útil y deben ser reemplazados. Otros, en cambio, han estado en desuso por años y necesitan ser repuestos, pero solo una docena de empresas en el mundo realizan el diseño del sistema y los costos para un bypass simple rondan las decenas de millones de dólares. Por esto, es importante contar con sistemas electrónicos de diseño y fabricación nacional.

Además, existen diferentes lugares donde aún resta instalar este tipo de sistemas, por lo que su implementación constituye una necesidad real para el desarrollo de la infraestructura ferroviaria de señalamiento en Argentina.

#### 1.5. Tipos de enclavamientos

#### 1.4. Sistema de enclavamientos 11

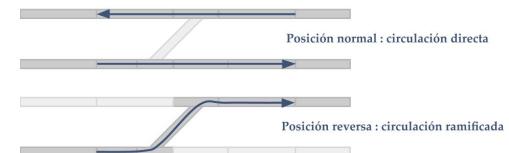


FIGURA 1.15: Posiciones normal e inversa del cambio.

#### 1.4. Sistema de enclavamientos

A modo de ejemplo se ilustra en la figura 1.16 un sistema de cambios en una vía simple con *bypass*. Este permite que dos formaciones puedan cruzarse en sentidos opuestos sin colisionar.

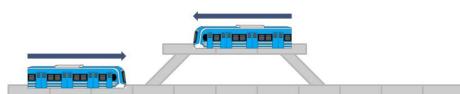


FIGURA 1.16: Vía simple con bypass.

Para evitar la colisión, se requiere un control seguro que evite que las formaciones avancen hacia secciones ya ocupadas por otras. También debe evitar que las formaciones avancen sobre los cambios cuando estos aún no han terminado de posicionarse en su lugar, lo que provocaría descarrillamientos. A este control se lo denomina sistema de enclavamiento y en definitiva impide que se produzcan las configuraciones no seguras y controla los semáforos que habilitan o no los itinerarios de las formaciones.

Una falla en un enclavamiento puede poner en peligro cientos de vidas humanas y generar gastos considerables. Por lo tanto, en el diseño del sistema de enclavamiento se deben cumplir estrictos parámetros de fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS).

Lamentablemente, los sistemas de enclavamientos en Argentina son en su mayoría mecánicos, de comienzos del siglo XX, y otra cantidad considerable son electromecánicos, de más de 40 años de antigüedad. Muchos de ellos ya han agotado su vida útil y deben ser reemplazados. Otros, en cambio, han estado en desuso por años y necesitan ser repuestos, pero solo una docena de empresas en el mundo realizan el diseño del sistema y los costos para un bypass simple rondan las decenas de millones de dólares. Por esto, es importante contar con sistemas electrónicos de diseño y fabricación nacional.

Además, existen diferentes lugares donde aún resta instalar este tipo de sistemas, por lo que su implementación constituye una necesidad real para el desarrollo de

#### 12

#### Capítulo 1. Introducción General

la infraestructura ferroviaria de señalamiento en Argentina.

#### 1.5. Tipos de enclavamientos

A continuación se presentan distintos tipos y etapas de implementación de enclavamientos.

A continuación se presentan distintas tecnologías de implementación de enclavamientos en orden cronológico de invención.

### 1.5.1. Enclavamientos mecánicos

A comienzos del siglo XX se implementaron los sistemas de enclavamientos mediante soluciones mecánicas. Utilizaban palancas como las que se visualizan en la figura 1.14 para comandar los cambios de vías y semáforos.

Una vez que se constituye una configuración de posiciones de palancas que habilitan un trayecto, estas quedan "enclavadas" mecánicamente. Es decir, su posición se bloquea y no es físicamente posible cambiarla. A medida que se van moviendo ciertas palancas, las demás que pudieran representar situaciones no seguras quedan enclavadas, y solo se pueden mover aquellas cuyo accionamiento representa

#### 1.5. Tipos de enclavamientos

11



FIGURA 1.14: Sistema mecánico de enclavamiento en la antigua estación de Chascomús, hoy convertida en museo.

una situación segura. De esa manera se garantiza que no se generarán configuraciones tales que las formaciones colisionen entre sí.

Las tecnologías más modernas heredaron el término "enclavamiento", aunque ya no se tengan palancas enclavadas en posiciones fijas.

### 1.5.2. Enclavamientos electromecánicos

A mediados del siglo XX se desarrolló el sistema de enclavamiento electromecánico. Su funcionamiento se basa en relés (figura 1.15) y circuitos de vía, de forma tal de poder detectar la presencia de un tren y comandar tanto las señales como las barreras de los pasos a nivel.



FIGURA 1.15: Bastidor de relés de estación Lavallol (Línea Roca).

12

#### Capítulo 1. Introducción General

Los sistemas de enclavamiento electromecánicos son comandados por un operario mediante un panel de control (figura 1.16). El operario solicita al sistema de enclavamiento las rutas que el conductor ferroviario necesita para circular. El sistema permitirá solo la operación de cambio de vías seguras. En caso contrario, se tendrán las salidas "enclavadas" y el sistema de enclavamiento impedirá mediante los semáforos el avance de la formación hasta que pueda realizarse el cambio en forma segura.

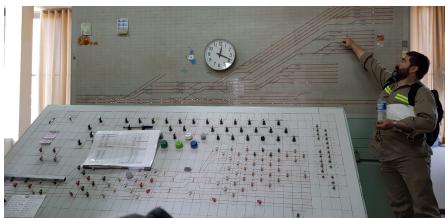


FIGURA 1.16: Panel de control enclavamientos - Central Lavallol.

### 1.5.3. Enclavamientos electrónicos

El sistema de enclavamiento moderno es electrónico y debe incluir redundancia de hardware para lograr niveles RAMS adecuados. Pueden utilizarse, por ejemplo, estrategias de 2 en 2 o sistemas de votación 2 de 3, como se ve en la figura 1.17, para tener siempre una salida sin importar los fallos que puedan producirse[7]. La diversidad de plataformas de hardware (representados en diferente color en la figura 1.17), por ejemplo mediante diferentes proveedores de un mismo sistema o plataforma, es otra técnica muy utilizada.

A continuación se presentan distintas tecnologías de implementación de enclavamientos en orden cronológico de invención.

### 1.5.1. Enclavamientos mecánicos

A comienzos del siglo XX se implementaron los sistemas de enclavamientos mediante soluciones mecánicas. Utilizaban palancas como las que se visualizan en la figura 1.17 para comandar los cambios de vías y semáforos.



FIGURA 1.17: Sistema enclavamiento mecánico en la estación de Chascomús, hoy convertida en museo.

Una vez que se constituye una configuración de posiciones de palancas que habilitan un trayecto, estas quedan "enclavadas" mecánicamente. Es decir, su posición se bloquea y no es físicamente posible cambiarla. A medida que se van moviendo ciertas palancas, las demás que pudieran representar situaciones no seguras quedan enclavadas, y solo se pueden mover aquellas cuyo accionamiento representa una situación segura. De esa manera se garantiza que no se generarán configuraciones tales que las formaciones colisionen entre sí.

Las tecnologías más modernas heredaron el término "enclavamiento", aunque ya no se tengan palancas enclavadas en posiciones fijas.

### 1.5.2. Enclavamientos electromecánicos

A mediados del siglo XX se desarrolló el sistema de enclavamiento electromecánico. Su funcionamiento se basa en relés (figura 1.18) y circuitos de vía, de forma tal de poder detectar la presencia de un tren y comandar tanto las señales como las barreras de los pasos a nivel.

#### 1.5. Tipos de enclavamientos

13



FIGURA 1.18: Bastidor de relés de estación Lavallol (Línea Roca).

Los sistemas de enclavamiento electromecánicos son comandados por un operario mediante un panel de control (figura 1.19). El operario solicita al sistema de enclavamiento las rutas que el conductor ferroviario necesita para circular. El sistema permitirá solo la operación de cambio de vías seguras. En caso contrario, se tendrán las salidas "enclavadas" y el sistema de enclavamiento impedirá mediante los semáforos el avance de la formación hasta que pueda realizarse el cambio en forma segura.

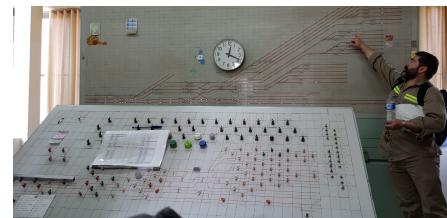


FIGURA 1.19: Panel de control enclavamientos - Central Lavallol.

### 1.5.3. Enclavamientos electrónicos

El sistema de enclavamiento moderno es electrónico y debe incluir redundancia de hardware para lograr niveles RAMS adecuados. La redundancia en sistemas críticos se define por medio de la abreviatura NooM, donde M representa a la cantidad de módulos de medición o decisión que posee el sistema y N la cantidad de dichos módulos que deben funcionar correctamente para que el sistema

14

#### Capítulo 1. Introducción General

opere normalmente[22]. De la investigación realizada surge que el 66% de las empresas utiliza una redundancia 2oo2 o 2oo3 para alcanzar los niveles de seguridad requeridos[4, 8, 13, 18]. Sólo una pequeña porción de las mismas utiliza

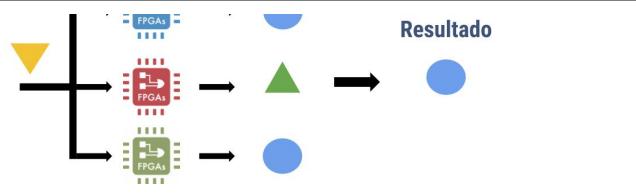


FIGURA 1.17: Redundancia por votación 2 de 3.

En este trabajo se implementó un sistema de enclavamiento electrónico, como se explicará en el Capítulo 3.

## 1.6. Objetivos

13

### 1.6. Objetivos

El objetivo de este proyecto fue el diseño, implementación y realización de pruebas funcionales de un prototipo de sistema electrónico de enclavamiento, sobre un kit de desarrollo de FPGA.

Se procuró además estudiar las tecnologías para implementar metodologías orientadas a mejorar los niveles RAMS del sistema, de acuerdo con el estado del arte en sistemas ferroviarios altamente críticos.

Teniendo la experiencia acumulada del trabajo realizado en la Especialización de Sistemas Embebidos, se puso especial énfasis en la automatización del proceso, para poder satisfacer las necesidades de cualquier zona elegida.

tauna redundancia 2002[6] o 2004[7].

En consecuencia, se puede afirmar que una redundancia 2002 o 2003 (figura 1.20) es representativa de los sistemas analizados y puede utilizarse como esquema de partida para un diseño propio. En un sistema con redundancia 2003 se tendrá una salida correcta siempre que se tenga a lo sumo un fallo simultáneo[23].

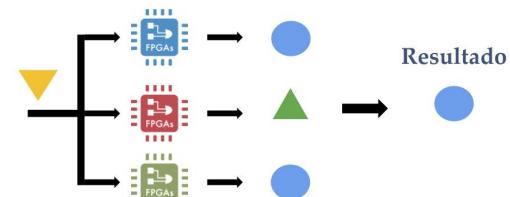


FIGURA 1.20: Redundancia por votación 2 de 3.

También se ilustra en la Figura 1.20 el concepto de diversidad de plataformas de hardware (representados en diferente color). Para mitigar fallos comunes a una misma plataforma de hardware es una buena estrategia el utilizar sistemas de diferentes marcas o proveedores, para minimizar esta problemática. De esta forma, el resultado global es un sistema inmune a fallas singulares. No obstante, es vulnerable a fallas simultáneas de dos componentes, pero su probabilidad es mínima al ser de diferentes tecnologías u orígenes.

En este trabajo se implementó un sistema de enclavamiento electrónico, como se explicará en el Capítulo 3.

## 1.6. Objetivos

El objetivo de este proyecto fue el diseño, implementación y realización de pruebas funcionales de un prototipo de sistema electrónico de enclavamiento, sobre un kit de desarrollo de FPGA.

Se procuró además estudiar las tecnologías para implementar metodologías orientadas a mejorar los niveles RAMS del sistema, de acuerdo con el estado del arte en sistemas ferroviarios altamente críticos.

Teniendo la experiencia acumulada del trabajo realizado en la Especialización de Sistemas Embebidos, se puso especial énfasis en la automatización del proceso, para poder satisfacer las necesidades de cualquier zona elegida.

## Capítulo 2

### Introducción Específica

Es este capítulo se presentan las topologías **básicas** del sistema ferroviario y los dos enfoques de resolución del proyecto, con sus ventajas y desventajas antes de abordar la implementación de la solución elegida.

#### 2.1. Topologías básicas

El tendido ferroviario argentino dista de ser uniforme: en las zonas rurales predominan vías simples con bypass debido al alto costo de utilizar vías dobles, mientras que en las zonas urbanas son mayoría las estaciones y playas de maniobras a talleres de mantenimiento.

Es por eso que el trabajo realizado debe abocarse a **muchos frentes** y muy diversos unos de otros. La cantidad de elementos involucrados **diverge** conforme la complejidad de la topología **se incrementa**, pero los elementos y sus comportamientos no dejan de ser los mismos de los presentados en el capítulo 1.

##### 2.1.1. Bypass

Para cubrir grandes distancias entre un punto estratégico, como podría ser Vaca Muerta, y un puerto para la exportación de los recursos se requiere una línea ferroviaria con vagones de carga que puedan transportar el producto. Por supuesto, es necesario un ida y vuelta entre trenes vacíos que van a ser llenados en el destino y trenes llenos que van a descargar al puerto la carga que lleven, pero el costo de utilizar dos vías en sentidos opuestos es muy elevado y usar solo una no tiene el menor sentido logístico.

Es por eso que se utiliza la topología de bypass cada cierta cantidad de kilómetros de vías simples para poder permitir que trenes en sentidos opuestos se crucen sin riesgo de colisión. En la figura 2.1 se representa la topología bypass, donde un tren que circula hacia la derecha por la vía principal podría entrar en colisión lateral con un tren que busca ingresar a la red desde una vía secundaria por medio del cambio de vías.

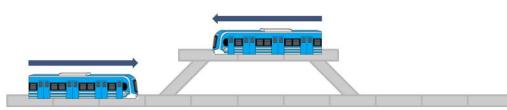
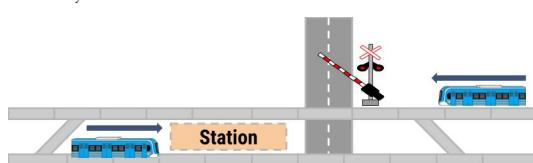


FIGURA 2.1: Topología bypass.

##### 2.1.2. Estación

Los lugares donde se habilita a los pasajeros a ingresar o salir del tren se denominan andenes y se encuentran en las estaciones ferroviarias situadas cada cierta cantidad de kilómetros, en diferentes localidades del país. En la figura 2.2 se muestra una topología de estación simple con andén central. Es decir, el mismo andén permite utilizar trenes tanto de la vía ascendente como de la descendente. Se añadió un paso a nivel vehicular en las inmediaciones de la estación y dos cambios de vías en orientaciones opuestas para permitir a la **hipotética estación** realizar trayectos cortos entre terminales.



## Capítulo 2

### Introducción Específica

Es este capítulo se presentan las topologías **típicas** del sistema ferroviario y los enfoques de resolución, con sus ventajas y desventajas antes de abordar la implementación de la solución elegida.

#### 2.1. Topologías típicas

El tendido ferroviario argentino dista de ser uniforme: en las zonas rurales predominan vías simples con bypass debido al alto costo de utilizar vías dobles, mientras que en las zonas urbanas son mayoría las estaciones y playas de maniobras a talleres de mantenimiento.

Es por eso que el trabajo realizado debe abocarse a **muchas locaciones** y muy diversos unos de otros. La cantidad de elementos involucrados **incrementa** conforme la complejidad de la topología **aumenta**, pero los elementos y sus comportamientos no dejan de ser los mismos de los presentados en el capítulo 1.

##### 2.1.1. Bypass

Para cubrir grandes distancias entre un punto estratégico, como podría ser Vaca Muerta, y un puerto para la exportación de los recursos se requiere una línea ferroviaria con vagones de carga que puedan transportar el producto. Por supuesto, es necesario un ida y vuelta entre trenes vacíos que van a ser llenados en el destino y trenes llenos que van a descargar al puerto la carga que lleven, pero el costo de utilizar dos vías en sentidos opuestos es muy elevado y usar solo una no tiene el menor sentido logístico.

Es por eso que se utiliza la topología de bypass cada cierta cantidad de kilómetros de vías simples para poder permitir que trenes en sentidos opuestos se crucen sin riesgo de colisión. En la figura 2.1 se representa la topología bypass, donde un tren que circula hacia la derecha por la vía principal podría entrar en colisión lateral con un tren que busca ingresar a la red desde una vía secundaria por medio del cambio de vías.

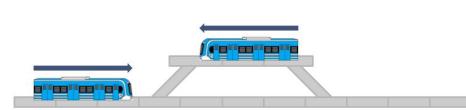


FIGURA 2.1: Topología bypass.

##### 2.1.2. Estación

Los lugares donde se habilita a los pasajeros a ingresar o salir del tren se denominan andenes y se encuentran en las estaciones ferroviarias situadas cada cierta cantidad de kilómetros, en diferentes localidades del país. En la figura 2.2 se muestra una topología de estación simple con andén central. Es decir, el mismo andén permite utilizar trenes tanto de la vía ascendente como de la descendente. Se añadió un paso a nivel vehicular en las inmediaciones de la estación y dos cambios de vías en orientaciones opuestas para permitir a la **formación** realizar trayectos cortos entre terminales.





FIGURA 2.2: Topología estación con andén único.

Aunque estaciones de andén único como la estación de Gerli de la Línea Roca son comunes, también existen estaciones con dos andenes: uno puramente para utilizar la vía ascendente y alejarse de la terminal, y otro puramente descendente para viajar hacia la terminal. Tal es el caso de estaciones como Longchamps, Adrogue, Remedios de Escalada de la Línea Roca.

### 2.1.3. Hub

Otras topologías más complejas incluyen una cantidad extra de andenes, como Lomas de Zamora o Temperley, que pueden recibir formaciones de varios ramales distintos (Glew/A. Korn, Ezeiza, Bosques, Quilmes, La Plata, etc.) o incluso tener accesos a playas de maniobras o talleres de reparación para poder injectar o retirar formaciones de la red. Tal es el caso representado en la figura 2.3, donde



FIGURA 2.2: Topología estación con andén único.

Aunque estaciones de andén único como la estación de Gerli de la Línea Roca son comunes, también existen estaciones con dos andenes: uno puramente para utilizar la vía ascendente y alejarse de la terminal, y otro puramente descendente para viajar hacia la terminal. Tal es el caso de estaciones como Longchamps, Adrogue, Remedios de Escalada de la Línea Roca.

### 2.1.3. Hub

Otras topologías más complejas incluyen una cantidad extra de andenes, como Lomas de Zamora o Temperley, que pueden recibir formaciones de varios ramales distintos (Glew/A. Korn, Ezeiza, Bosques, Quilmes, La Plata, etc.) o incluso tener accesos a playas de maniobras o talleres de reparación para poder injectar o retirar formaciones de la red. Tal es el caso representado en la figura 2.3, donde se tiene una mayor cantidad de andenes, ingresos/egresos a talleres ferroviarios

## 2.1. Topologías típicas

17

se tiene una mayor cantidad de andenes, ingresos/egresos a talleres ferroviarios o bifurcaciones a otros ramales, además de la rama principal de circulación entre terminales.

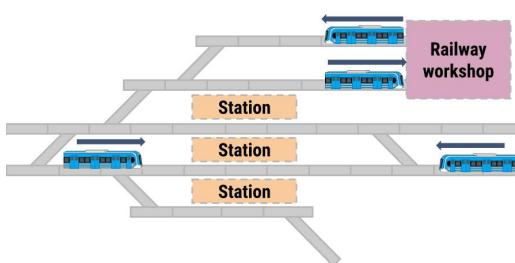


FIGURA 2.3: Topología hub.

Un buen ejemplo de esta topología es la estación Llavallol de la Línea Roca. En ella se tiene una extensa playa de maniobras como la comandada por el panel de control de la figura 1.16.

### 2.1.4. Terminal

Finalmente, la topología más compleja de todas es la denominada terminal, tal como se representa en la figura 2.4. En ella se encuentran numerosos andenes que pueden ser tanto de ingreso como de egreso indistintamente, presentan muchos cambios de vías para facilitar el intercambio de formaciones y permitir que las vías funcionen en ambos sentidos de circulación, además de tener ramificaciones para diversos ramales que convergen en la terminal.

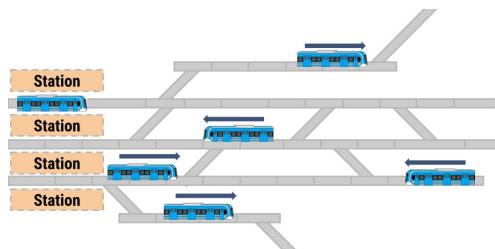


FIGURA 2.4: Topología terminal.

## 2.1. Topologías típicas

17

o bifurcaciones a otros ramales, además de la rama principal de circulación entre terminales.

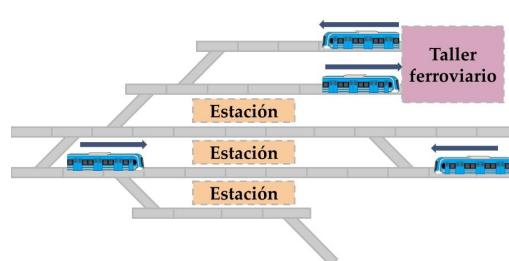


FIGURA 2.3: Topología hub.

Un buen ejemplo de esta topología es la estación Llavallol de la Línea Roca. En ella se tiene una extensa playa de maniobras como la comandada por el panel de control de la figura 1.19.

### 2.1.4. Terminal

Finalmente, la topología más compleja de todas es la denominada terminal, tal como se representa en la figura 2.4. En ella se encuentran numerosos andenes que pueden ser tanto de ingreso como de egreso indistintamente, presentan muchos cambios de vías para facilitar el intercambio de formaciones y permitir que las vías funcionen en ambos sentidos de circulación, además de tener ramificaciones para diversos ramales que convergen en la terminal.

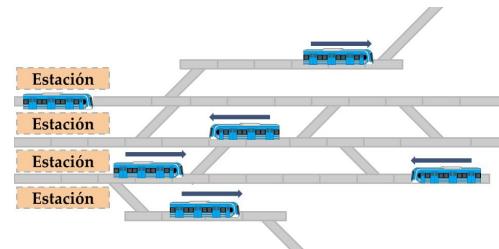


FIGURA 2.4: Topología terminal.

Ejemplos de esta topología pueden ser las estaciones Once de septiembre (Línea Sarmiento), Alejandro Korn y Constitución (Línea Roca) o Retiro (Línea San Martín).

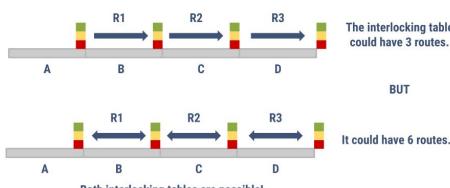
18

## Capítulo 2. Introducción Específica

Ejemplos de esta topología pueden ser las estaciones Once de septiembre (Línea Sarmiento), Alejandro Korn y Constitución (Línea Roca) o Retiro (Línea San Martín, Línea Mitre, Línea Belgrano Norte y Cargas). Incluso hay estaciones que, aunque no son el final del recorrido, pueden funcionar como terminales de ramales más cortos tales como Ezeiza, que extiende el ramal Constitución-Ezeiza hasta Cañuelas, o Glew, que extiende el ramal Constitución-Glew hasta Alejandro Korn.

### 2.2. Enfoque funcional

A la hora de definir itinerarios se utiliza el concepto de ruta, que es el camino definido entre dos semáforos consecutivos. Pero no siempre se establecen todas las rutas posibles, sino solo las necesarias para generar los itinerarios. Por ejemplo, en la figura 2.5 se pueden ver dos casos de definición de rutas para una misma topología de vía simple.



18

## Capítulo 2. Introducción Específica

Martín, Línea Mitre, Línea Belgrano Norte y Cargas). Incluso hay estaciones que, aunque no son el final del recorrido, pueden funcionar como terminales de ramales más cortos tales como Ezeiza, que extiende el ramal Constitución-Ezeiza hasta Cañuelas, o Glew, que extiende el ramal Constitución-Glew hasta Alejandro Korn.

### 2.2. Estrategias de resolución

A lo largo de la investigación se elevaron decenas de artículos, los cuales abordan el problema con dos estrategias de desarrollo muy diferenciadas, con sus ventajas y desventajas: el enfoque funcional y el enfoque geográfico.

En el enfoque funcional, el desarrollo se basa en la denominada «tabla de enclavamientos». En la misma cada fila define una ruta que el enclavamiento debe poder ejecutar en caso de cumplir las condiciones que define expresamente cada una. Cada itinerario se forma a partir de la conjunción de diferentes rutas, siempre que estas sean compatibles, lo cual también está indicado en la tabla.

Históricamente, tanto los enclavamientos mecánicos como los electromecánicos se han definido previamente por tablas de enclavamientos, las cuales también pueden ser usadas para definir la logística de la red. Es por eso que el personal técnico está muy capacitado tanto en la lectura de la tabla como en su elaboración. Automatizar el proceso es inmediato ya que la tabla de enclavamientos define todas las funcionalidades necesarias.

En el enfoque geográfico, en cambio, no hay una descripción macro del sistema a nivel funcional, sino una descripción a nivel de componentes. Es la interacción

FIGURA 2.5: Ejemplo de elaboración de las rutas.

En el primer caso se **puede querer utilizar la vía en un solo sentido**, para lo cual el semáforo A es el  **inicial de la ruta R<sub>1</sub>** y el semáforo B es el **final de esta**. Luego se **definen las otras dos rutas y el itinerario para recorrer toda la topología es la concatenación de las rutas R<sub>1</sub>, R<sub>2</sub> y R<sub>3</sub>**, como se describe en la Tabla 2.1.

TABLA 2.1: Tabla de enclavamientos (Caso unidireccional)

Ruta	Señal de entrada	Señal de salida
1	Semáforo <sub>A</sub>	Semáforo <sub>B</sub>
2	Semáforo <sub>B</sub>	Semáforo <sub>C</sub>
3	Semáforo <sub>C</sub>	Semáforo <sub>D</sub>

En el segundo caso de la figura 2.5 el **itinerario deberá contemplar un uso bidireccional de la vía**, por lo que se **añaden** las rutas R<sub>4</sub>, R<sub>5</sub> y R<sub>6</sub>, como se describe en la Tabla 2.2.

entre los componentes, en función de su posición geográfica en el entorno, lo que se manifiesta a nivel macro. De poder definir las reglas genéricas de cada elemento y las condiciones en las cuales interactúan, es posible definir cualquier sistema, sin tabla de enclavamientos.

Este enfoque dista de ser inmediato, ya que es necesario construir todas las herramientas de análisis de la red y diseñar cada elemento al detalle. La ventaja es que, concluida esa etapa, es un enfoque flexible y completamente independiente de la localización. Además de definir todas las funcionalidades que la red admite y no solo las necesarias para la logística.

En las siguientes secciones se profundizará en ambos enfoques, los conceptos necesarios para su plena comprensión, cómo se modifica el flujo de trabajo y el análisis de su escalabilidad. Esto último es de vital importancia a la hora de elegir uno de los dos enfoques.

### 2.3. Enfoque funcional

El enfoque funcional necesita de un elemento central para su desarrollo, sin el cuál no es posible definir la funcionalidad del sistema: la tabla de enclavamientos.

#### 2.3.1. Tabla de enclavamientos

A la hora de establecer itinerarios se utiliza el concepto de ruta, que es el camino definido entre dos semáforos consecutivos. Pero no siempre se establecen

### 2.3. Enfoque funcional

19

todas las rutas posibles, sino solo las necesarias para generar los itinerarios. Por ejemplo, en la figura 2.5 se pueden ver dos casos de definición de rutas para una misma topología de bypass.

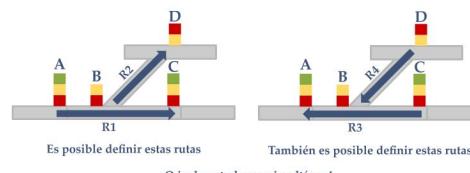


FIGURA 2.5: Ejemplo de elaboración de las rutas.

En el primer caso se **utilizarán las vías en un solo sentido (izquierda a derecha)**, para lo cual el semáforo A es el  **inicial de la ruta R<sub>1</sub>** y el semáforo C es el **final de esta**. Luego se define la ruta R<sub>2</sub> entre los semáforos de maniobra B y D para **completar todas las posibles rutas con sentido izquierda a derecha**, como se describe en la Tabla 2.1.

TABLA 2.1: Tabla de enclavamientos (izquierda a derecha)

Ruta	Señal de entrada	Señal de salida
1	Semáforo <sub>A</sub>	Semáforo <sub>C</sub>
2	Semáforo <sub>B</sub>	Semáforo <sub>D</sub>

En el segundo caso de la figura 2.5 el **itinerario podría contemplar también un uso estrictamente de derecha a izquierda**. Sumando así las rutas R<sub>3</sub> y R<sub>4</sub>, como se describe en la Tabla 2.2.

TABLA 2.2: Tabla de enclavamientos (derecha a izquierda)

Ruta	Señal de entrada	Señal de salida
3	Semáforo <sub>C</sub>	Semáforo <sub>A</sub>
4	Semáforo <sub>D</sub>	Semáforo <sub>B</sub>

Según el uso que se le quiera dar a la topología, ambas tablas de enclavamientos son válidas. Incluso podemos notar que ambas están incompletas porque una tabla no contempla los casos de la otra. Podríamos definir una nueva tabla de enclavamientos como la conjunción de ambas, como se describe en la Tabla 2.3.

Se añade a la tabla el concepto de ruta conflictiva. Es decir, rutas que no pueden coexistir con la ruta que se quiere pedir. Por ejemplo, R<sub>1</sub> y R<sub>3</sub> van en sentidos opuestos y por lo tanto no puede permitirse R<sub>1</sub> si un tren se encuentra realizando R<sub>3</sub>. De la misma forma, como R<sub>1</sub> requiere que la máquina de cambios se encuentre en posición normal, no puede coexistir ni con R<sub>3</sub> ni con R<sub>4</sub> que requieren el que el cambio se encuentre en posición reversa. Es posible deducir el resto de rutas conflictivas de forma análoga.

20

Capítulo 2. Introducción Específica

TABLA 2.3: Tabla de enclavamientos (caso bidireccional)

Ruta	Señal de entrada	Señal de salida	Rutas conflictivas
1	Semáforo <sub>A</sub>	Semáforo <sub>C</sub>	R <sub>2</sub> , R <sub>3</sub> y R <sub>4</sub>
2	Semáforo <sub>B</sub>	Semáforo <sub>D</sub>	R <sub>1</sub> , R <sub>3</sub> y R <sub>4</sub>
3	Semáforo <sub>C</sub>	Semáforo <sub>A</sub>	R <sub>1</sub> , R <sub>2</sub> y R <sub>4</sub>
4	Semáforo <sub>D</sub>	Semáforo <sub>B</sub>	R <sub>1</sub> , R <sub>2</sub> y R <sub>3</sub>

Tanto la Tabla 2.1 como la 2.2 son una porción de la llamada tabla de enclavamientos, que se utiliza para diseñar los sistemas de enclavamientos tanto mecánicos como electromecánicos y define completamente el comportamiento del sistema. En esta, cada ruta constituye una fila y presenta diferentes columnas tales como:

- Semáforos de entrada y de salida.
- Circuitos de vías que deben estar desocupados para permitir la ruta.
- Pasos a nivel que deben tener la barrera baja para permitir la ruta.
- Posición del cambio requerida para permitir la ruta.
- Rutas conflictivas que inhiben la activación de la ruta.

Como se pudo ver en ambos ejemplos, la necesidad de tales o cuales itinerarios puede requerir distintas tablas de enclavamientos para la misma topología. Al-

20

## Capítulo 2. Introducción Específica

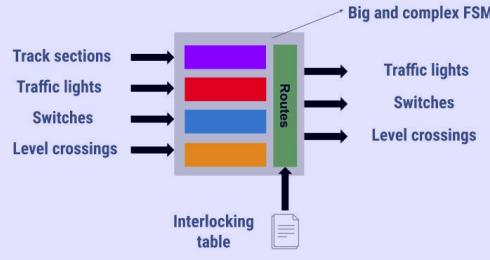


FIGURA 2.6: Enfoque funcional.

La salida del sistema actúa sobre todo el señalamiento, menos la ocupación de los tramos de vías porque son elementos de solo lectura. Puede verse que conforme se añadan más rutas a la tabla de enclavamientos, las máquinas de estados serán más complejas y de mayor tamaño.

## 2.2.2. Flujo de trabajo

La tabla de enclavamientos es la piedra angular de todo el proceso, sin ella no se puede realizar ningún diseño. En la figura 2.7 se ilustra el flujo de trabajo en el enfoque funcional.

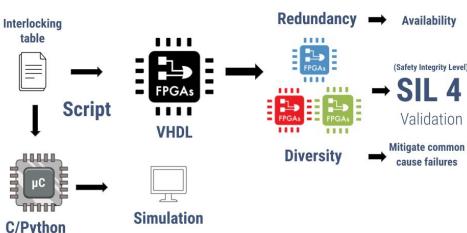


FIGURA 2.7: Esquema de trabajo en el enfoque funcional.

Como bien se resaltó anteriormente, el diseño nace de la tabla de enclavamientos, y para evitar que cualquier error en esta se propague al sistema, la tabla es

## 2.2. Enfoque funcional

21

diseñada y revisada por varios profesionales ferroviarios. Pero si aun así no es suficiente, en la UTN-Haedo se ha avanzado en el desarrollo de un *script* que busca errores en las tablas de enclavamientos.

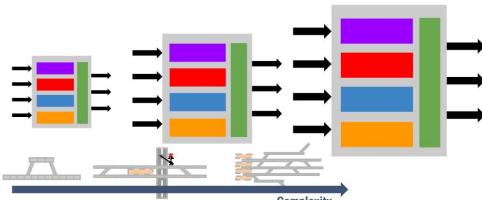
El proceso culmina con la redundancia de los sistemas diseñados y la diversificación de plataformas, sin lo cual no se podrían alcanzar estándares de calidad y seguridad altos como los demandados por industrias tan críticas como la nuclear, la aeronáutica, etc.

En el transcurso de la Especialización de Sistemas Embebidos se trabajó en un diseño con enfoque funcional de la estación Belgrano R de la Línea Mitre, pero sin automatización del proceso. En él, la tabla de enclavamientos fue una pieza vital de todo el desarrollo.

Durante 2019 se avanzó en la automatización de este flujo de trabajo, que culminó con la presentación de un artículo en el Congreso Argentino de Sistemas Embebidos 2019 y en una edición especial de IEEE Latin American Transaction.

## 2.2.3. Escalabilidad de la estrategia

Al ir automatizando el sistema se fue llegando a la conclusión de que los bloques que contenían las máquinas de estados crecían en tamaño hasta volverse inmanejables las conexiones. En la figura 2.8 se representa el concepto del crecimiento del sistema conforme la topología se vuelve más compleja.



gunas tablas serán más completas que otras y eso puede repercutir en que, al querer añadir nuevas rutas a futuro, la tabla deba ser modificada y por lo tanto el desarrollo del sistema deba cambiar a otro más complejo.

## 2.3.2. Modelo del sistema

En el enfoque funcional se utiliza la tabla de enclavamientos como elemento central de decisión para el diseño y funcionamiento del sistema. Es la ruta la que impone qué acciones deben ser tomadas y cuáles prohibidas, y por lo tanto se abstrae de la topología una vez definida la tabla. En la figura 2.6 se presenta un modelo del sistema con enfoque funcional.

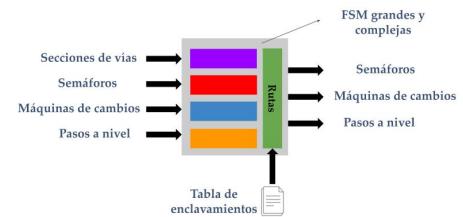


FIGURA 2.6: Enfoque funcional.

## 2.3. Enfoque funcional

21

Cada bloque horizontal en diferentes colores representa máquinas de estados que modelan los elementos de entrada indicados. Todos ellos, gobernados por una máquina de estados general representada en un bloque vertical verde que se divide en base a la tabla de enclavamientos.

La salida del sistema actúa sobre todo el señalamiento, menos la ocupación de los tramos de vías porque son elementos de solo lectura. Puede verse que conforme se añaden más rutas a la tabla de enclavamientos, las máquinas de estados serán más complejas y de mayor tamaño.

## 2.3.3. Flujo de trabajo

En el enfoque funcional, la tabla de enclavamientos es la piedra angular de todo el proceso, sin ella no se puede realizar ningún diseño. En la figura 2.7 se ilustra el flujo de trabajo para este enfoque.

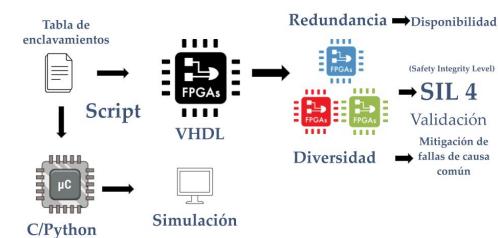


FIGURA 2.7: Esquema de trabajo en el enfoque funcional.

Como bien se resaltó anteriormente, el diseño nace de la tabla de enclavamientos, y para evitar que cualquier error en esta se propague al sistema, la tabla es diseñada y revisada por varios profesionales ferroviarios. La validación de las tablas puede ser automatizada mediante *scripts* que busquen errores en las mismas.

El proceso culmina con la redundancia de los sistemas diseñados y la diversificación de plataformas, sin lo cual no se podrían alcanzar estándares de calidad y seguridad altos como los demandados por industrias tan críticas como la nuclear, la aeronáutica, etc.

En el transcurso de la Especialización de Sistemas Embebidos se trabajó en un diseño con enfoque funcional de la estación Belgrano R de la Línea Mitre, pero sin automatización del proceso. En él, la tabla de enclavamientos fue una pieza vital de todo el desarrollo.

## 2.3.4. Escalabilidad de la estrategia

Al ir automatizando la generación del código del sistema se fue llegando a la conclusión de que los bloques que contenían las máquinas de estados crecían en

22

## Capítulo 2. Introducción Específica

tamaño hasta volverse inmanejables las conexiones. En la figura 2.8 se representa el concepto del crecimiento del sistema conforme la topología se vuelve más compleja.

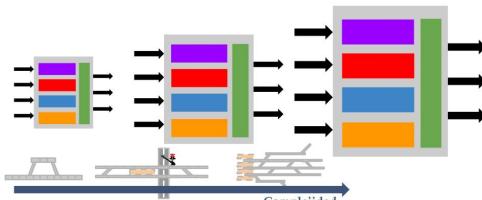


FIGURA 2.8: Escalabilidad del enfoque funcional.

Es destacable que a medida que la topología se complejiza la cantidad de bloques para modelarla sigue siendo la misma. Lo que se incrementa es la cantidad de estados en cada bloque y la densidad de conexiones tanto entre estos como internamente.

El tener bloques monolíticos y de crecimiento exponencial perjudica el desarrollo de los tests, que deben ser desechados y reescritos de cero cada vez que la topología cambia, por mas mínimos que sean esos cambios.

Otro problema encontrado es el de la incompletitud, que se detalló al inicio de la Sección 2.2. Si la red admite  $M$  rutas pero solo se definieron  $N$  como necesarias (donde  $N \leq M$ ), entonces se podrían tener  $N$  tests que las validen y el sistema se certifica para  $N$  rutas. Pero si en un futuro se necesitan  $N+1$  rutas, se deberá hacer todo el proceso de validación desde cero, lo que encarece todo el proyecto.

22

## Capítulo 2. Introducción Específica

La ventaja central de este enfoque es que, teniendo la tabla de enclavamientos correctamente definida, es muy sencillo e inmediato pasar de la tabla a la implementación del sistema. Aunque el uso de memoria es excesivo y desde el punto de vista del testing y la validación es incompleto.

**2.3. Enfoque geográfico**

En el mundo de la electrónica el concepto de red circuitual se usa ampliamente. El funcionamiento de la red a nivel macro es la consecuencia directa de los principios físicos de funcionamiento de pequeños elementos discretos (resistores, capacitores, inductores, etc.) y su interconexión entre ellos. Sabiendo como funciona un componente es posible predecir y simular el comportamiento de una red entera llena de estos, sin importar que sean de diferente valor; solo basta con que sean regidos por el mismo principio físico.

Es entonces que removemos el concepto de ruta como elemento central del desarrollo y nos enfocamos a modelar los elementos discretos de la red ferroviaria (semáforos, barreras, cambios, circuitos de vía). Este enfoque se denomina geográfico, ya que el comportamiento macro de la red es consecuencia directa de la relación entre los elementos mencionados. En la figura 2.9 se representa la idea central de este enfoque: la variedad de elementos es finita y puede ser modelada fácilmente; el foco del desarrollo debe estar centrado en el análisis de la red ferroviaria misma.

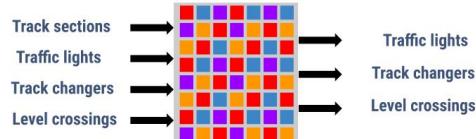


FIGURA 2.9: Enfoque geográfico.

Se puede apreciar en la figura que la variedad de colores de los pequeños bloques es limitada. Esto se debe a que cada uno simboliza un elemento ferroviario, sin importar cuál sea. El concepto a destacar es que una vez que se ha modelado un bloque rojo, todos los bloques rojos se comportarán igual y es su posición relativa a los otros bloques (sean o no rojos) lo que tendrá una funcionalidad a nivel general.

**2.3.1. Análisis de grafos**

Un grafo es una representación matemática de las relaciones (aristas) entre componentes (nodos) de una red. Utilizado ampliamente en ciencias de la computación y matemática, es sencillo llegar a un modelo de grafos partiendo desde una topología como la de la figura 2.10, donde a cada tramo de vía se le asignó un nodo y cada arista del grafo representa el vínculo que existe entre ese tramo y sus vecinos.

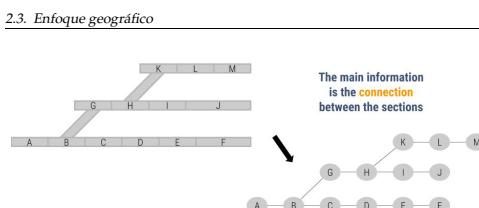


FIGURA 2.10: Pasaje de topología ferroviaria a grafo.

Por ejemplo, el nodo B posee un cambio de vías que vincula los tramos A y C si el cambio se encuentra en posición normal y los tramos A y G si el cambio se encuentra en posición inversa. Por lo tanto, en el grafo, el nodo B posee aristas que lo vinculan con los nodos A,C y G.

La misma topología de la figura 2.10 puede ser analizada por el algoritmo creado para este trabajo utilizando como información únicamente el grafo que la modela. Todos los nodos que tengan un solo vecino son extremos de la red, mientras que los que tengan tres vecinos se asumirá que poseen un cambio de vías. Los nodos que tengan dos vecinos serán analizados según su posición relativa a cambios cercanos. El resultado de analizar esta topología se muestra en la figura 2.11.

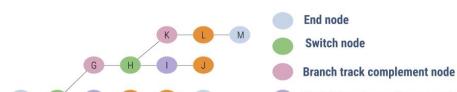


FIGURA 2.8: Escalabilidad del enfoque funcional.

Es importante destacar que a medida que la topología se complejiza la cantidad de bloques para modelarla sigue siendo la misma. Lo que se incrementa es la cantidad de estados en cada bloque y la densidad de conexiones tanto entre estos como internamente.

El tener bloques monolíticos y de crecimiento exponencial perjudica el desarrollo de los tests, que deben ser desechados y reescritos por completo cada vez que la topología cambia, por mas mínimos que sean esos cambios.

Otro problema encontrado es el de la incompletitud, que se detalló al inicio de la Sección 2.3. Si la red admite  $M$  rutas pero solo se definieron  $N$  como necesarias (donde  $N \leq M$ ), entonces se podrían tener  $N$  tests que las validen y el sistema se certifica para  $N$  rutas. Pero si en un futuro se necesitan  $N+1$  rutas, se deberá hacer todo el proceso de validación desde cero, lo que encarece todo el proyecto.

La ventaja central de este enfoque es que, teniendo la tabla de enclavamientos correctamente definida, es muy sencillo e inmediato pasar de la tabla a la implementación del sistema. La desventaja es que el uso de memoria es excesivo y desde el punto de vista del testing y la validación es incompleto.

**2.4. Enfoque geográfico**

En el enfoque geográfico el énfasis no está puesto en la funcionalidad a nivel sistema sino en la interacción entre los componentes dependiendo de la posición de los mismos en la red. Por ende no necesita una tabla de enclavamientos que defina su comportamiento, sino definir genéricamente los componentes y establecer una representación matemática de las conexiones entre ellos. En ese sentido, este enfoque requiere un nivel de análisis previo mucho mayor y, por lo tanto, un tiempo de desarrollo más elevado. No obstante, su flexibilidad a la hora de escalar a topologías de mayor tamaño es notable.

Es entonces que removemos el concepto de ruta como elemento central del desarrollo y nos enfocamos a modelar los elementos discretos de la red ferroviaria (semáforos, barreras, cambios, circuitos de vía). En la figura 2.9 se representa la idea central de este enfoque: la variedad de elementos es finita y puede ser modelada fácilmente; el foco del desarrollo debe estar centrado en el análisis de la red ferroviaria misma.

## 2.4. Enfoque geográfico

23

idea central de este enfoque: la variedad de elementos es finita y puede ser modelada fácilmente; el foco del desarrollo debe estar centrado en el análisis de la red ferroviaria misma.

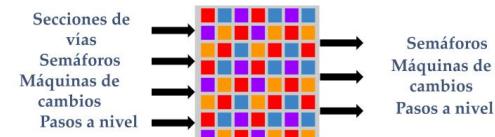


FIGURA 2.9: Enfoque geográfico.

Se puede apreciar en la figura 2.9 que la variedad de colores de los pequeños bloques es limitada. Esto se debe a que cada uno simboliza un elemento ferroviario, sin importar cuál sea. El concepto a destacar es que una vez que se ha modelado un bloque rojo, todos los bloques rojos se comportarán igual y es su posición relativa a los otros bloques (sean o no rojos) lo que tendrá una funcionalidad a nivel general.

**2.4.1. Análisis de grafos**

Para poder representar las conexiones entre los elementos ferroviarios, se introduce el concepto de grafo. Un grafo es una representación matemática de las relaciones (aristas) entre componentes (nodos) de una red. Utilizado ampliamente en ciencias de la computación y matemática, es sencillo llegar a un modelo de grafos partiendo desde una topología como la de la figura 2.10, donde a cada tramo de vía se le asignó un nodo y cada arista del grafo representa el vínculo que existe entre ese tramo y sus vecinos.

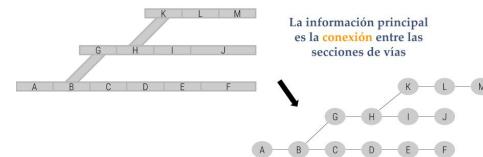


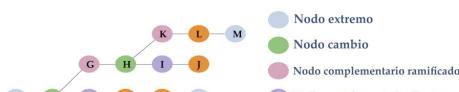
FIGURA 2.10: Pasaje de topología ferroviaria a grafo.

Por ejemplo, el nodo B posee un cambio de vías que vincula los tramos A y C si el cambio se encuentra en posición normal y los tramos A y G si el cambio se encuentra en posición inversa. Por lo tanto, en el grafo, el nodo B posee aristas que lo vinculan con los nodos A,C y G.

24

## Capítulo 2. Introducción Específica

La misma topología de la figura 2.10 puede ser analizada por el algoritmo creado para este trabajo utilizando como información únicamente el grafo que la modela. Todos los nodos que tengan un solo vecino son extremos de la red, mientras que los que tengan tres vecinos se asumirá que poseen un cambio de vías. Los nodos que tengan dos vecinos serán analizados según su posición relativa a cambios cercanos. El resultado de analizar esta topología se muestra en la figura 2.11.





Nodos como el G y el K deben analizarse por su posición relativa a los nodos B y H, que son la raíz del cambio. Al desprendese de la rama principal de circulación son categorizados como complementos de rama. En cambio los nodos C e I, al continuar el trayecto que tienen los nodos A-B y G-H, son complementos directos de la rama, porque la extienden más allá de los segmentos indicados.

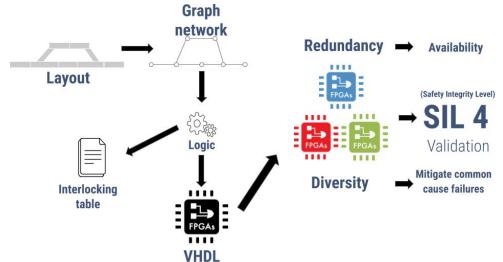
Otros nodos como D,E y L no tienen ninguna característica especial en este ejemplo, pero bien podrían categorizarse de otra manera si por los tramos de vías que representan se tuviese un paso a nivel.

### 2.3.2. Flujo de trabajo

El procedimiento se puede repetir infinitamente para cualquier topología, ya que sin importar la cantidad de elementos o sus conexiones todos los vínculos entre componentes pueden ser representados mediante un grafo. En la figura 2.12 se ilustra el esquema de trabajo seguido para el enfoque geográfico.

24

Capítulo 2. Introducción Específica



A diferencia del enfoque funcional, en el enfoque geográfico la tabla de enclavamientos ocupa un rol secundario al ser un historial del proceso de conversión entre el gráfico y la implementación electrónica del sistema. Obviamente la tabla de enclavamientos del enfoque funcional (al ser incompleta) debe estar contenida en la tabla de enclavamientos del enfoque geográfico (al considerar todas las rutas que admite la red). Ambos enfoques, aunque parten de conceptos distintos, deben converger en los mismos resultados y ser consistentes a la hora de comparar ambas tablas.

El proceso se inicia con el pasaje, de momento manual, del layout al grafo. Este es analizado por el algoritmo que detecta cuántos semáforos, de cuántos aspectos, en qué orientación y dónde deben situarse para que el sistema sea seguro. Esto, además de detectar la posición de todos los cambios y barreras, encontrando todas las rutas soportadas por la red y generando una tabla de enclavamientos completa.

El proceso culmina de forma idéntica al enfoque funcional: aplicando estrategias de redundancia y diversidad se buscará alcanzar un nivel de seguridad alto propio de la industria nuclear o aeroespacial.

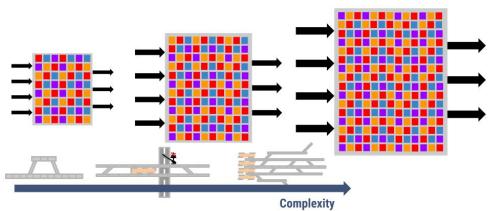
### 2.3.3. Escalabilidad de la estrategia

Realizando el mismo análisis de escalabilidad se llega a una conclusión muy diferente respecto del otro enfoque. Al aumentar la complejidad, y por lo tanto el tamaño, de las topologías, el tamaño de los bloques se mantiene constante, pero se incrementa la cantidad de bloques necesarios para implementar el sistema. Esto es ilustrado en la figura 2.13.

Todos los tests unitarios, referidos a cada bloque del mismo color que modelan un mismo espacio físico (que puede o no contener una barrera, un cambio o varios semáforos), se elaboran una única vez, sin importar la cantidad de elementos idénticos presentes en la topología. Sumado a que, como en este enfoque se identifican todas las rutas y por lo tanto se pueden generar todos los tests necesarios, la batería de ensayos que otorga este enfoque es completa. Es decir, siempre se

2.3. Enfoque geográfico

25



tendrá una cantidad de tests mayor o igual que la necesaria para cualquier necesidad presente o futura.

Una desventaja de este enfoque es que se debe implementar el analizador de redes ferroviarias y un conversor que a partir de un grafo genere toda la estructura de archivos necesaria para implementar el circuito electrónico en una FPGA. Por lo tanto, la complejidad y, en consecuencia, el tiempo de desarrollo son mayores.



Nodos como el G y el K deben analizarse por su posición relativa a los nodos B y H, que son la raíz del cambio. Al desprendese de la rama principal de circulación son categorizados como complementos de rama. En cambio los nodos C e I, al continuar el trayecto que tienen los nodos A-B y G-H, son complementos directos de la rama, porque la extienden más allá de los segmentos indicados.

Otros nodos como D,E y L no tienen ninguna característica especial en este ejemplo, pero bien podrían categorizarse de otra manera si por los tramos de vías que representan se tuviese un paso a nivel.

### 2.4.2. Flujo de trabajo

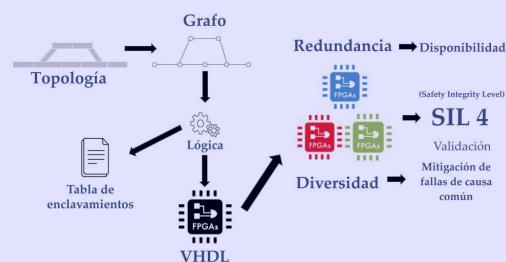
El procedimiento se puede repetir infinitamente para cualquier topología, ya que sin importar la cantidad de elementos o sus conexiones todos los vínculos entre componentes pueden ser representados mediante un grafo. En la figura 2.12 se ilustra el esquema de trabajo seguido para el enfoque geográfico.

A diferencia del enfoque funcional, en el enfoque geográfico la tabla de enclavamientos ocupa un rol secundario al ser un historial del proceso de conversión entre el gráfico y la implementación electrónica del sistema. Es **immediato deducir** que la tabla de enclavamientos del enfoque funcional (al ser incompleta) debe estar contenida en la tabla de enclavamientos del enfoque geográfico (al considerar todas las rutas que admite la red). Ambos enfoques, aunque parten de conceptos distintos, deben converger en los mismos resultados y ser consistentes a la hora de comparar ambas tablas.

El proceso se inicia con el pasaje, de momento manual, del layout al grafo. Este es analizado por el algoritmo que detecta cuántos semáforos, de cuántos aspectos, en qué orientación y dónde deben situarse para que el sistema sea seguro. Esto, además de detectar la posición de todos los cambios y barreras, encontrando todas las rutas soportadas por la red y generando una tabla de enclavamientos completa.

### 2.4. Enfoque geográfico

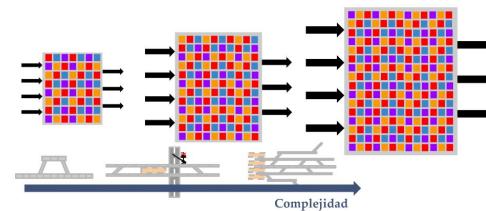
25



El proceso culmina de forma idéntica al enfoque funcional: aplicando estrategias de redundancia y diversidad se buscará alcanzar un nivel de seguridad alto propio de la industria nuclear o aeroespacial.

### 2.4.3. Escalabilidad de la estrategia

Realizando el mismo análisis de escalabilidad se llega a una conclusión muy diferente respecto del otro enfoque. Al aumentar la complejidad, y por lo tanto el tamaño, de las topologías, el tamaño de los bloques se mantiene constante, pero se incrementa la cantidad de bloques necesarios para implementar el sistema. Esto es ilustrado en la figura 2.13.



Todos los tests unitarios, referidos a cada bloque del mismo color que modelan un mismo espacio físico (que puede o no contener una barrera, un cambio o varios semáforos), se elaboran una única vez, sin importar la cantidad de elementos idénticos presentes en la topología. Sumado a que, como en este enfoque se identifican todas las rutas y por lo tanto se pueden generar todos los tests necesarios, la batería de ensayos que otorga este enfoque es completa. Es decir, siempre se

26

Capítulo 2. Introducción Específica

tendrá una cantidad de tests mayor o igual que la necesaria para cualquier necesidad presente o futura.

Una desventaja de este enfoque es que se debe implementar el analizador de redes ferroviarias y un conversor que a partir de un grafo genere toda la estructura de archivos necesaria para implementar el circuito electrónico en una FPGA. Por lo tanto, la complejidad y, en consecuencia, el tiempo de desarrollo son mayores.

### 2.5. Consideraciones generales

En el presente trabajo se optó por implementar el sistema bajo el enfoque funcio-

En este trabajo se optó por implementar el sistema bajo el enfoque funcional, asumiendo que sus ventajas son mucho más fuertes que sus desventajas y el análisis de los resultados obtenidos fueron mucho más provechosos que los de su contraparte funcional.

Los módulos del sistema fueron implementados con máquinas de estado finitas con camino de datos (FSMD, del inglés *Finite State Machine with Data path*), que son máquinas de estado finitas (FSM, del inglés *Finite State Machine*) y circuitos secuenciales. La FSMD (figura 2.14) posee dos partes diferenciadas: el camino de control y el camino de datos. El camino de control contiene una FSM que, según las entradas de control y el estado interno que posee, genera señales internas que controlan los circuitos secuenciales del camino de datos. Estos, a su vez, contienen los bloques que procesan las entradas y actúan sobre las salidas.

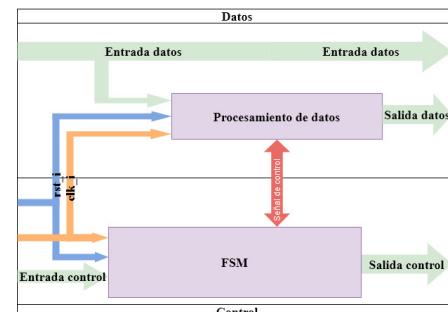


FIGURA 2.14: Diagrama en bloques genérico de una FSMD

Siguiendo los lineamientos recomendados, una FSMD debe ser diseñada, implementada y simulada de acuerdo con los siguientes pasos:

1. Definición del algoritmo a implementar.
2. Definición de entradas y salidas de la FSMD.
3. Diseño del camino de datos.

## Capítulo 3

### Diseño e Implementación

En este capítulo se presentan las decisiones de diseño adoptadas para concretar el desarrollo del trabajo. Además, se describen en forma genérica los módulos necesarios tanto del sistema de enclavamiento como de los bloques auxiliares para concretar una comunicación exitosa entre el sistema y el exterior.

#### 3.1. Consideraciones generales

En el presente trabajo se optó por implementar el sistema bajo el enfoque funcional, asumiendo que sus ventajas son mucho más fuertes que sus desventajas y el análisis de los resultados obtenidos fueron mucho más provechosos que los de su contraparte funcional.

Los módulos del sistema fueron implementados con máquinas de estado finitas con camino de datos (FSMD, del inglés *Finite State Machine with Data path*), que son máquinas de estado finitas (FSM, del inglés *Finite State Machine*) y circuitos secuenciales. La FSMD (figura 3.1) posee dos partes diferenciadas: el camino de control y el camino de datos. El camino de control contiene una FSM que, según las entradas de control y el estado interno que posee, genera señales internas que controlan los circuitos secuenciales del camino de datos. Estos, a su vez, contienen los bloques que procesan las entradas y actúan sobre las salidas.

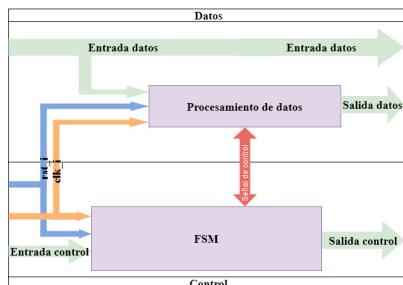


FIGURA 3.1: Diagrama en bloques genérico de una FSMD

Siguiendo los lineamientos recomendados, una FSMD debe ser diseñada, implementada y simulada de acuerdo con los siguientes pasos:

1. Definición del algoritmo a implementar.
2. Definición de entradas y salidas de la FSMD.
3. Diseño del camino de datos.
4. Diseño de interfaz entre camino de datos y camino de control.
5. Definición de los estados de la FSM.
6. Diseño de la FSM.
7. Implementación del diseño.
8. Diseño e implementación de los ensayos.

Esta metodología puede inferir mas tiempo de desarrollo que el habitual, pero ya ha demostrado ser exitosa en el proyecto realizado por el Mg. Ing. Facundo Larosa, codirector de este trabajo. Por lo que se aprovechó su experiencia y conocimiento para resolver esta etapa del desarrollo. Los beneficios son un mayor control del diseño a bajo nivel, una mayor portabilidad y un mas eficiente uso de los recursos de la plataforma electrónica.

### 3.2. Plataforma utilizada

Por razones de disponibilidad se utilizó el kit de desarrollo Arty Z7 (figura 3.2), el cual posee 17600 LUT's, 35200 FF's, 32 BUFG's y 100 IOB's [cite28]. Se lo utilizó como base para sintetizar el diseño y extraer conclusiones que permitan dimensionar los recursos lógicos necesarios para un desarrollo de estas características.

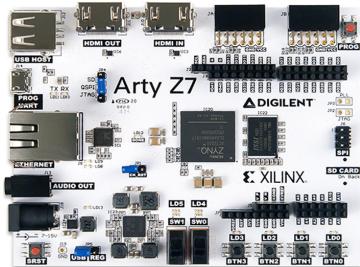


FIGURA 3.2: FPGA Arty Z7-10 de Digilent

### 2.6. Plataforma utilizada

4. Diseño de interfaz entre camino de datos y camino de control.
5. Definición de los estados de la FSM.
6. Diseño de la FSM.
7. Implementación del diseño.
8. Diseño e implementación de los ensayos.

Esta metodología puede inferir mas tiempo de desarrollo que el habitual, pero ya ha demostrado ser exitosa en el proyecto realizado por el Mg. Ing. Facundo Larosa, codirector de este trabajo. Por lo que se aprovechó su experiencia y conocimiento para resolver esta etapa del desarrollo. Los beneficios son un mayor control del diseño a bajo nivel, una mayor portabilidad y un mas eficiente uso de los recursos de la plataforma electrónica.

### 2.6. Plataforma utilizada

Se utilizó una plataforma FPGA, a diferencia de las empresas mencionadas en el capítulo 1 que utilizan microprocesadores, ya que se busca aprovechar la concurrencia y mayor seguridad inherentes a esta tecnología[24-26] para la implementación de sistemas críticos.

Por razones de disponibilidad se utilizó el kit de desarrollo Arty Z7 (figura 2.15), el cual posee 17600 LUT's, 35200 FF's, 32 BUFG's y 100 IOB's [27]. Se lo utilizó como base para sintetizar el diseño y extraer conclusiones que permitan dimensionar los recursos lógicos necesarios para un desarrollo de estas características.

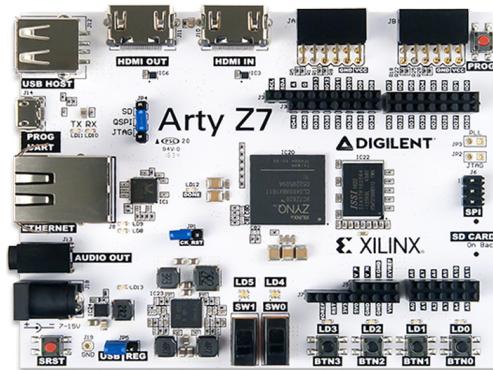


FIGURA 2.15: FPGA Arty Z7-10 de Digilent

Capítulo 3

## Diseño e Implementación

En este capítulo se presentan las decisiones de diseño adoptadas para concretar el desarrollo del trabajo. Además, se describen en forma genérica los módulos necesarios tanto del sistema de esclavamiento como de los bloques auxiliares para concretar una comunicación exitosa entre el sistema y el exterior.

### 3.1. Analizador de redes ferroviarias

En el marco de la Especialización de Sistemas Embebidos se utilizó el enfoque funcional basado en la tabla de enclavamientos, con las limitaciones ya expuestas en el capítulo 2. La mayoría de los artículos encontrados utilizan ese enfoque[28-31], pero una pequeña porción de los artículos aborda el modelado de los sistemas sin utilizar tablas de enclavamiento [32-35].

Los diversos artículos que modelan los sistemas ferroviarios en base a la topología utilizan estrategias muy dispares. Por ejemplo, no existe una idea en común de que herramienta matemática utilizar: algunos utilizan redes de Petri, mientras que otros utilizan grafos. Incluso dentro del subgrupo de los artículos que sugieren utilizar grafos no tienen un criterio unificado de qué representa cada nodo y cada arista, ni tampoco cómo analizar la red ni cuáles son los elementos básicos que necesita para estar definida.

En base a la bibliografía relevada, se llegó a la conclusión de que todo grafo ferroviario necesita dos datos para estar definido. El primero es la lista de relaciones entre nodo inicial y nodo final, y el segundo es la posición absoluta del nodo en el grafo junto con datos adicionales, como si posee un paso a nivel o si es bidireccional.

Con esa información fue posible realizar un analizador de redes ferroviarias. Un script implementado en python que procesa archivos de texto plano donde se indican las conexiones entre ellos y cuyos resultados son:

- Análisis de red ferroviaria:
    - Determina qué cantidad de semáforos son necesarios.
    - Determina dónde deben situarse los semáforos.
    - Determina cuántos aspectos deben tener los semáforos.
    - Determina qué orientación deben tener los semáforos.
    - Identifica todos las máquinas de cambios.

### 3.3. Análisis de la red ferroviaria y generación automática del código

### **3.3. Análisis de la red ferroviaria y generación automática del código**

Todo grafo ferroviario necesita dos datos para estar definido. El primero es la lista de relaciones entre nodo inicial y nodo final, y el segundo es la posición absoluta del nodo en el grafo junto con datos adicionales, como si posee un paso a nivel o no es bidireccional. Con esa información es posible realizar un análisis para obtener un resultado como el que se presenta en la figura 3.3, al ingresar un grafo de una red ferroviaria bypass.

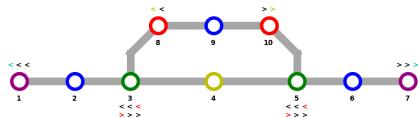


FIGURA 3.3: Grafo luego de ser analizado por el algoritmo.

Los nodos 1 y 7 se encuentran pintados de violeta porque, al tener un único vecino cada uno, se consideran nodos extremos. Los nodos 2, 6 y 9 no presentan nada especial, por lo que son nodos simples. Los que sí tienen una importancia central en el análisis son los nodos que poseen tres vecinos: el nodo 3 y el nodo 5, que están pintados en verde y se consideran cambios raíz.

Luego el nodo 4 se categoriza como nodo de cambio directo por ser la continuación de los segmentos 2-3 y 5-6 de tener el cambio de vía en posición normal permitiendo la circulación directa. En este caso el nodo 4 es compartido por ambos cambios, pero no siempre se da este caso.

Los nodos 8 y 10 que son vecinos de los cambios 3 y 5 pero no comparten ninguna coordenada espacial con ellos, son nodos de cambios ramificados. Es decir solo permitirán la secuencia de nodos 2-3-8 o 9-8-3 si la máquina de cambios se encuentra en posición inversa. Para el nodo 5 el análisis es análogo.

La asignación de semáforos se realiza solo sobre los nodos extremos, cambios raíz y cambios ramificados. Los extremos necesitan los semáforos para permitir la salida de las formaciones de la red, ya que la red ferroviaria continúa más allá del nodo 1 y del 7; de ser nodos extremos absolutos (fin de red) no corresponde que se les asigne un semáforo.

Los nodos de cambios son los que presentan mayor cantidad de semáforos. Necesitan dos semáforos de tres aspectos para permitir la circulación directa sobre el cambio cuando se encuentra en posición normal y un semáforo de dos aspectos para permitir la circulación en la ramificación del recorrido, pero con precaución por ser una zona crítica. Por último, los nodos de cambios ramificados solo presentan un semáforo de doble aspecto como complemento al otro semáforo de maniobras, en función de utilizar la ramificación para volver al recorrido principal a una velocidad moderada.

- Identifica los extremos de la red, sean absolutos o relativos.
  - Identifica todas las rutas soportadas por la red.
  - Genera una tabla de enclavamientos con todos los datos obtenidos.

■ Implementación de la red ferroviaria en VHDL:

  - Genera todos los archivos necesarios.
  - Interconecta todos los módulos creados.
  - Adapta el tamaño de todas las señales a las que la topología necesite.

■ Interfaz de comunicación con la red ferroviaria:

- Genera todos las tramas necesarias para la comunicación.
- Brinda un menú de opciones para modificar las tramas en tiempo real.
- Envía las tramas al sistema implementado en la FPGA.
- Actualiza la interfaz con los datos devueltos por la FPGA.

Todas las topologías de la sección 2.1 pueden ser analizadas por el programa y generar, de forma automática, todo el código en VHDL que sea necesario para el correcto funcionamiento del sistema. A modo de ejemplo se incluye el caso de la figura 3.1, producto de ingresar un grafo de una red ferroviaria bypass.

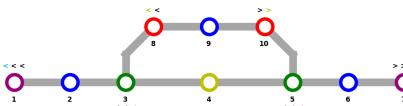


FIGURA 3.1: Grafo luego de ser analizado por el algoritmo

Los nodos 1 y 7 se encuentran pintados de violeta porque, al tener un único vecino cada uno, se consideran nodos extremos. Los nodos 2, 6 y 9 no presentan nada en especial, por lo que son nodos simples. Los que sí tienen una importancia central en el análisis son los nodos que poseen tres vecinos: el nodo 3 y el nodo 5, que están pintados en verde y se consideran cambios raíz.

Luego el nodo 4 se categoriza como nodo de cambio directo por ser la continuación de los segmentos 2-3 y 5-6 de tener el cambio de vía en posición normal, permitiendo la circulación directa. En este caso el nodo 4 es compartido por ambos cambios, pero no siempre se da este caso.

Los nodos 8 y 10, que son vecinos de los cambios 3 y 5 pero no comparten ninguna coordenada espacial con ellos, son nodos de cambios ramificados. Es decir, solo permitirán la secuencia de nodos 2-3-8 o 9-8-3 si la máquina de cambios se encuentra en posición inversa. Para el nodo 5 el análisis es análogo.

La asignación de semáforos se realiza solo sobre los nodos extremos, cambios raíz y cambios ramificados. Los extremos necesitan los semáforos para permitir la salida de las formaciones de la red, ya que la red ferroviaria continúa más allá

### 3.4. Módulo de nodos

Como existen diversos tipos de nodos instanciados en el sistema, se describirá el funcionamiento de un nodo genérico con la máxima cantidad de funcionalidades, de forma tal de cubrir todos los casos.

Un módulo de nodo (cuyo diagrama de bloques se presenta en la figura 3.4) recibe los estados de ocupación de sus vecinos y de si mismo desde el exterior, además del estado de los semáforos que posee.

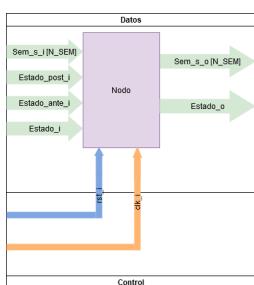


FIGURA 3.4: FSMD del módulo de nodo genérico

Internamente deberá informar su estado de ocupación a sus vecinos y decidir los aspectos que deberán tener sus semáforos de la siguiente manera:

- Si el tramo propio está ocupado: el semáforo propio estará en aspecto rojo.
- Si el vecino está ocupado: el semáforo propio estará en aspecto rojo.
- Si el vecino está desocupado:
  - Si el semáforo vecino está en rojo: el semáforo analizado estará en aspecto amarillo.
  - Si el semáforo vecino está en amarillo: el semáforo analizado estará en aspecto verde.

### 3.5. Módulo de la máquina de cambios cambios

Un cambio de vías en la realidad conecta un tramo A con un tramo B o con un tramo C. Pero en la implementación de los nodos solo se contemplan dos casos: un vecino anterior y un vecino posterior. Es por eso que el nodo de la máquina de cambios tiene como función el conectar con uno o otro nodo, como se muestra en la figura 3.5.

De esta manera, el nodo A ejemplificado en la figura 3.5 tendrá como vecino posterior al que le indique la máquina de cambios. Si la posición del cambio es normal, entonces el nodo A tendrá como vecino posterior al nodo B. Si la posición

### 3.2. Módulo de nodos

31

del nodo 1 y del 7; de ser nodos extremos absolutos (fin de red) no corresponde que se les asigne un semáforo.

Los nodos de cambios son los que presentan mayor cantidad de semáforos. Necesitan dos semáforos de tres aspectos para permitir la circulación directa sobre el cambio cuando se encuentra en posición normal y un semáforo de dos aspectos para permitir la circulación en la ramificación del recorrido, pero con precaución por ser una zona crítica. Por último, los nodos de cambios ramificados solo presentan un semáforo de doble aspecto como complemento al otro semáforo de mañobras, en función de utilizar la ramificación para volver al recorrido principal a una velocidad moderada.

El haber desarrollado un criterio propio para modelar las topologías tuvo como consecuencia el tener que diseñar tanto la herramienta de análisis, como así también la arquitectura y funcionalidad de cada uno de los módulos expuestos a continuación.

### 3.2. Módulo de nodos

El módulo principal a ser implementado es el módulo de nodos. Por cada nodo en el grafo se tendrá un módulo de nodos equivalente, cuyas conexiones a otros módulos estarán definidas por las aristas del grafo.

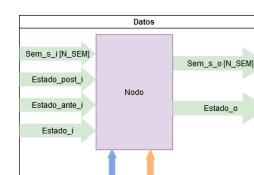
Se definió que cada nodo del grafo representa un tramo de vía, con todos los elementos que posee ese tramo en la realidad. Es decir, si el tramo incluye semáforos o barreras serán modeladas dentro del módulo de nodo. Con excepción de las máquinas de cambios que por su naturaleza tendrán un módulo aparte.

Como cada sección puede tener diferentes cantidades de elementos, existen diversos tipos de nodos en el sistema. Para exemplificar se describirá el funcionamiento de un nodo genérico con la máxima cantidad de funcionalidades, de forma tal de cubrir todos los casos.

Un módulo de nodo (cuyo diagrama de bloques se presenta en la figura 3.2) recibe los estados de ocupación de sus vecinos y de si mismo desde el exterior, además del estado de los semáforos que posee.

Internamente deberá informar su estado de ocupación a sus vecinos y decidir los aspectos que deberán tener sus semáforos de la siguiente manera:

- Si el tramo propio está ocupado: el semáforo propio estará en aspecto rojo.
- Si el vecino está ocupado: el semáforo propio estará en aspecto rojo.
- Si el vecino está desocupado:
  - Si el semáforo vecino está en rojo: el semáforo analizado estará en aspecto amarillo.
  - Si el semáforo vecino está en amarillo: el semáforo analizado estará en aspecto verde.



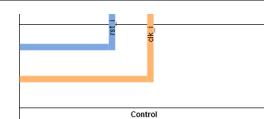


FIGURA 3.2: FSMD del módulo de nodo genérico

### 3.3. Módulo de la máquina de cambios

Una decisión de diseño importante fue que, a pesar de que los nodos del grafo pueden tener desde uno hasta seis vecinos, los módulos de nodos solo tienen dos posibles vecinos: el anterior y el posterior. Esto se hizo para que todos los nodos pudiesen ser modelados desde la misma plantilla base, agilizando la automatización del proceso.

Esto llevó a la necesidad de disponer de un módulo que commute las conexiones en el caso de tener nodos con más de dos vecinos. Este bloque es el módulo de la máquina de cambios.

Un cambio de vías conecta un tramo A con un tramo B o con un tramo C. El módulo de la máquina de cambios tiene como función el conectar un nodo anterior(A) con un nodo posterior(B) o un nodo del desvío(C), según la posición de la máquina de cambios, como se muestra en la figura 3.3.

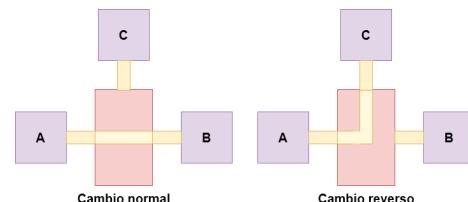


FIGURA 3.3: Conexión del módulo de la máquina de cambios

#### 3.6. Módulos de adaptación a enclavamiento

31

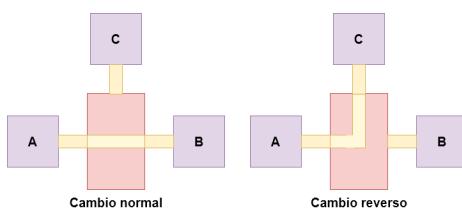


FIGURA 3.5: Conexión del módulo de la máquina de cambios

del cambio es reversa, entonces el nodo A tendrá como vecino posterior al nodo C.

De la misma manera los nodos B y C verán como nodo "anterior". El nodo A o a ningún nodo, según el cambio se encuentre en posición normal o reversa respectivamente.

En la figura 3.6 se ilustra el diagrama en bloques del módulo de la máquina de cambios.

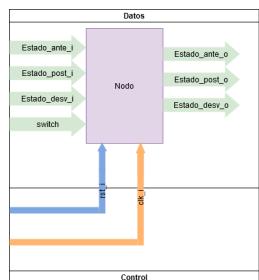


FIGURA 3.6: FSMD del módulo de máquina de cambios

#### 3.6. Módulos de adaptación a enclavamiento

El módulo de enclavamientos espera como entradas un vector de elementos booleanos de tamaño N y su salida será el mismo vector pero reducido a un tamaño M; por eso es necesario introducir dos módulos de adaptación. El primero debe separar el vector de N elementos booleanos (Tabla 3.1) en las señales que el enclavamiento necesita y el segundo debe recibir las señales de salida del sistema de enclavamiento (Tabla 3.2) y volver a generar el vector de elementos booleanos,

32

#### Capítulo 3. Diseño e Implementación

que al no incluir el estado de ocupación ahora tendrá un tamaño M menor que el original.

TABLA 3.1: Trama de datos a recibir

Trama de entrada [N]			
Ocupación	Semáforos	Barreras	Cambios

TABLA 3.2: Trama de datos a enviar

Trama de salida [M]		
Semáforos	Barreras	Cambios

La información de la ocupación tendrá tantos elementos como circuitos de vía a modelar, donde cada tramo libre se modelará con un 1 y cada ocupado con un 0. De igual forma para barreras y cambios.

La única diferencia es en el caso de los semáforos, donde la información de cada uno no ocupará 1 bit sino 2 bits, de forma tal de tener cuatro posibles aspectos:

#### 3.4. Módulos de adaptación a enclavamiento

33

De esta manera, el nodo A ejemplificado en la figura 3.3 tendrá como vecino posterior al que le indique la máquina de cambios. Si la posición del cambio es normal, entonces el nodo A tendrá como vecino posterior al nodo B. Si la posición del cambio es reversa, entonces el nodo A tendrá como vecino posterior al nodo C.

De la misma manera los nodos B y C verán como nodo "anterior" al nodo A o a ningún nodo, según el cambio se encuentre en posición normal o reversa respectivamente.

En la figura 3.4 se ilustra el diagrama en bloques del módulo de la máquina de cambios.

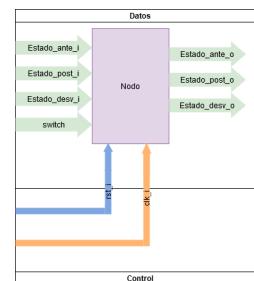


FIGURA 3.4: FSMD del módulo de máquina de cambios

#### 3.4. Módulos de adaptación a enclavamiento

Para el modelo de los estados de las barreras, circuitos de vías y máquinas se adoptó la convención de la Tabla 3.1.

TABLA 3.1: Convención para elementos ferroviarios I

Estado	Barrera	Máquina de cambios	Circuito de vía
'0'	Baja	Posición normal	Ocupado
'1'	Alta	Posición inversa	Desocupado

Però en el caso de los semáforos, dependiendo de la cantidad de aspectos, pueden tenerse tres o cuatro estados. Por lo tanto la convención que se definió es la que se puede ver en la Tabla 3.2.

Queda de manifiesto que podrían enviarse tramas conformadas únicamente por ceros y unos de forma serializada a la plataforma. Para luego procesar la trama e ir dividiendo cada porción en información a la zona que le corresponda.

Una forma de organizar la trama de entrada fue la adoptada en la Tabla 3.3. En la misma, se envían en orden todos los estados de ocupación, seguidos de todos

34

#### Capítulo 3. Diseño e Implementación

TABLA 3.2: Convención para elementos ferroviarios II

Estado	Semáforo [2 aspectos]	Semáforo [3 aspectos]	Semáforo [4 aspectos]
'00'	Rojo	Rojo	Rojo
'01'	Amarillo	Amarillo	Amarillo
'10'	■	■	Doble amarillo
'11'	■■	■■	■■■■

rojo (00), amarillo (01), doble amarillo (10) y verde (11).

### 3.6.1. Módulo separador

En la figura 3.7 se puede ver la FSMD del módulo separador. Este debe recibir el vector de elementos booleanos de tamaño N (paquete[N]) y la orden de que debe procesarlo (procesar).

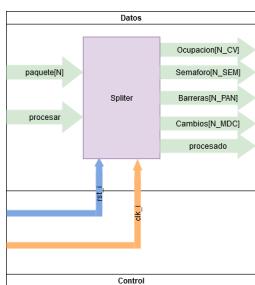


FIGURA 3.7: FSMD del módulo separador

A continuación, como el generador de código sabe previamente la cantidad de cada uno de los elementos ferroviarios, puede descomponer de forma sencilla los elementos del vector paquete[N] en vectores (o variables) más pequeñas según corresponda.

los estados de los semáforos (intercalando el bit más significativo con el menos significativo), los estados de las barreras y finalmente los estados de las máquinas de cambios.

TABLA 3.3: Trama de entrada [N]

Ocupación	Semáforos	Barreras	Cambios
-----------	-----------	----------	---------

La trama de salida será muy similar, salvo que los estados de ocupación no serán un dato a transmitir ya que son de solo lectura y se asume que el mismo no ha cambiado durante el tiempo de procesamiento. La trama se definió en la Tabla 3.4.

TABLA 3.4: Trama de salida [M]

Semáforos	Barreras	Cambios
-----------	----------	---------

El módulo de enclavamientos espera como entradas las señales de estado de cada elemento en paralelo. Pero, como ya definimos en las Tablas 3.3 y 3.4, la entrada del sistema es serializada. Por lo tanto, es necesario tener dos módulos que adapten ambas etapas:

- Módulo separador: encargado de convertir las entradas serializadas en señales en paralelo y distribuir los datos según donde sean requeridos.
- Módulo mediador: encargado de serializar las señales en paralelo que vienen del enclavamiento, según el orden requerido.

### 3.4.1. Módulo separador

El módulo separador diseñado se puede ver en la figura 3.5. Este debe recibir el vector de elementos booleanos de tamaño N ("paquete[N]") y la orden de que debe procesarlo ("procesar").

### 3.4. Módulos de adaptación a enclavamiento

35

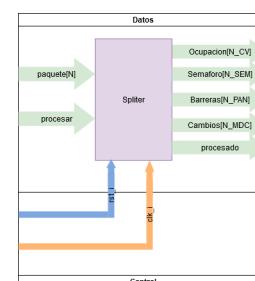


FIGURA 3.5: FSMD del módulo separador

A continuación, como el generador de código sabe previamente la cantidad de cada uno de los elementos ferroviarios, puede descomponer de forma sencilla los elementos del vector paquete[N] en vectores más pequeños según corresponda.

### 3.4.2. Módulo mediador

El módulo mediador diseñado, que se puede visualizar en la figura 3.6, tiene como función volver a generar el vector de elementos booleanos que ya han sido procesados por el enclavamiento.

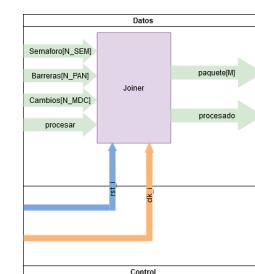


FIGURA 3.6: FSMD del módulo mediador

### 3.7. Módulos de procesamiento de tramas

33

#### 3.6.2. Módulo mediador

El bloque mediador que se puede visualizar en la figura 3.8 tiene como función volver a generar el vector de elementos booleanos que ya han sido procesados por el enclavamiento.

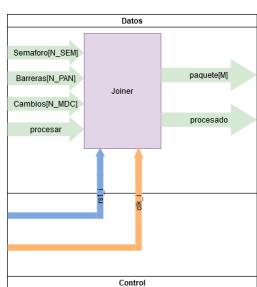


FIGURA 3.8: FSMD del módulo mediador

El módulo mediador recibe la salida del enclavamiento y la orden de generar el paquete ("procesar"). Una vez que el vector ("paquete[M]") ha sido creado, se envía una variable de control ("procesado") a la siguiente etapa para poder coordinar todo el sistema con un solo reloj.

### 3.7. Módulos de procesamiento de tramas

Durante el desarrollo del trabajo de la Especialización en Sistemas Embebidos se había considerado la estrategia de obtener las señales de forma paralela, es decir, para cada elemento se tenía asignado un pin que monitoreaba su estado. Esto resultó ser un problema a la hora de implementar topologías más grandes, donde se necesitó hasta cinco veces la cantidad de entradas digitales que la plataforma tenía. Por lo tanto, se decidió cambiar a una lectura y escritura en serie.

Pero el utilizar una lectura serie implica que debe indicarse cuál es el inicio y el final de cada mensaje, además de un criterio para determinar si el mensaje recibido es fiable.

#### 3.7.1. Módulo detector

36

### Capítulo 3. Diseño e Implementación

El módulo mediador recibe la salida del enclavamiento y la orden de generar el paquete ("procesar"). Una vez que el vector ("paquete[M]") ha sido creado, se envía una variable de control ("procesado") a la siguiente etapa para poder coordinar todo el sistema con un solo reloj.

El módulo detector tiene como función recibir una secuencia de caracteres y armar una salida con un vector de elementos booleanos. Un diagrama en bloques del funcionamiento del módulo se muestra en la figura 3.9.

34

## Capítulo 3. Diseño e Implementación

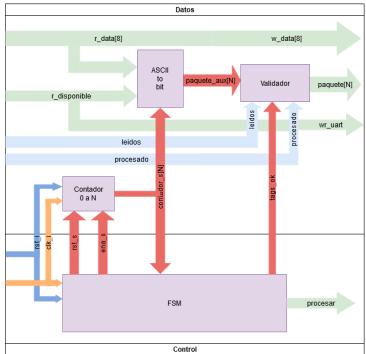


FIGURA 3.9: FSMD del módulo detector

La UART(del inglés, *Universal Asynchronous Receiver-Transmitter*) es la unidad encargada de recibir y transmitir las tramas desde la computadora hasta la plataforma. Esta envía secuencialmente un carácter por medio de la señal *r\_data* (8 bytes) y un pulso (*r\_disponible*) para informar que un nuevo dato ha sido enviado, además de indicar por medio de la señal *N* la cantidad de caracteres que serán enviados.

El proceso de detección es ilustrado en la figura 3.10.

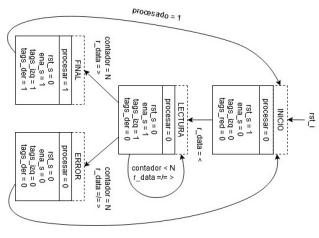


FIGURA 3.10: Estados del módulo detector

## 3.7. Módulos de procesamiento de tramas

35

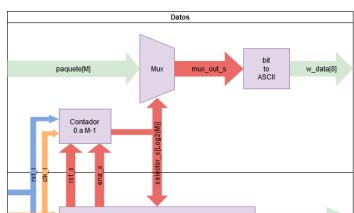
Se tiene un estado inicial en el cual se espera el carácter de inicio de la trama ("<") que provoca una transición al estado de lectura. En dicho estado se recibirán hasta N caracteres mientras se actualiza un contador interno. Cuando el contador interno iguale la cantidad N, se verifica si el próximo carácter es el de fin de trama (">").

Si el carácter leído es el de final de trama, se pasa al estado final, donde el paquete es considerado válido y enviado a la próxima etapa junto con su pulso de validación del dato. Si el carácter leído es distinto, entonces se descarta toda la trama y se vuelve al inicio a la espera de otro carácter de inicio de trama, reiniciando todas las variables auxiliares.

Internamente se tienen diversas variables auxiliares para controlar si se han recibido los delimitadores y si la cantidad recibida es correcta. Eso cobra gran importancia al realizar los ensayos, porque se puede diferenciar rápidamente la fuente de posibles errores.

## 3.7.2. Módulo registro

Así como el módulo de detección **realizaba** una conversión de caracteres (1 byte) a booleanos (1 bit), el módulo de registro (figura 3.11) hace la operación inversa. Dado un vector de elementos booleanos, el módulo debe generar M caracteres '0' o '1' según corresponda en base al vector, y enviarlos a la **UART** para su posterior impresión.



## 3.5. Módulos de procesamiento de tramas

Durante el desarrollo del trabajo de la Especialización en Sistemas Embebidos se había considerado la estrategia de obtener las señales de forma paralela, es decir, para cada elemento se tenía asignado un pin que monitoreaba su estado. Esto resultó ser un problema a la hora de implementar topologías más grandes, donde se necesitó hasta cinco veces la cantidad de entradas digitales que la **FPGA** tenía disponibles. Por lo tanto, se decidió cambiar a una lectura y escritura en serie.

Sin embargo, utilizar una lectura serie implica que debe indicarse cuál es el inicio y el final de cada mensaje, además de un criterio para determinar si el mensaje recibido es fiable. La solución desarrollada se presenta a continuación en esta sección.

## 3.5.1. Módulo detector

El módulo detector tiene como función recibir una secuencia de caracteres y armar una salida con un vector de elementos booleanos. Un diagrama en bloques del funcionamiento del módulo se muestra en la figura 3.7.

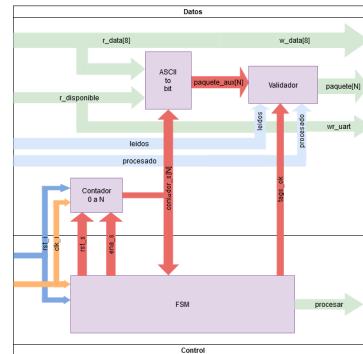


FIGURA 3.7: FSMD del módulo detector

## 3.5. Módulos de procesamiento de tramas

37

La UART(del inglés, *Universal Asynchronous Receiver-Transmitter*) es la unidad encargada de recibir y transmitir las tramas desde la computadora hasta la plataforma. Esta envía secuencialmente un carácter por medio de la señal *r\_data* (8 bytes) y un pulso (*r\_disponible*) para informar que un nuevo dato ha sido enviado, además de indicar por medio de la señal *N* la cantidad de caracteres que serán enviados.

El proceso de detección es ilustrado en la figura 3.8.

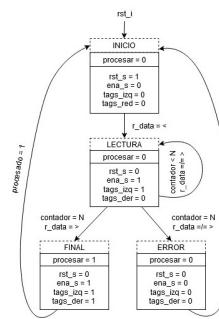


FIGURA 3.8: Estados del módulo detector

En la figura 3.8 se tiene un estado inicial en el cual se espera el carácter de inicio de la trama ("<") que provoca una transición al estado de lectura. En dicho estado se recibirán hasta N caracteres mientras se actualiza un contador interno. Cuando el contador interno iguale la cantidad N, se verifica si el próximo carácter es el de fin de trama (">").

Si el carácter leído es el de final de trama, se pasa al estado final, donde el paquete es considerado válido y enviado a la próxima etapa junto con su pulso de validación del dato. Si el carácter leído es distinto, entonces se descarta toda la trama y se vuelve al inicio a la espera de otro carácter de inicio de trama, reiniciando todas las variables auxiliares.

Internamente se tienen diversas variables auxiliares para controlar si se han recibido los delimitadores y si la cantidad recibida es correcta. Eso cobra gran importancia al realizar los ensayos, porque se puede diferenciar rápidamente la fuente de posibles errores.

## 3.5.2. Módulo registro

Así como el módulo de detección **realizaba** una conversión de caracteres (1 byte) a booleanos (1 bit), el módulo de registro (figura 3.9) hace la operación inversa. Dado un vector de elementos booleanos, el módulo debe generar M caracteres '0' o '1' según corresponda en base al vector, y enviarlos a la **UART** para su posterior impresión.

38

## Capítulo 3. Diseño e Implementación

o '1' según corresponda en base al vector, y enviarlos a la **UART** para su posterior impresión.

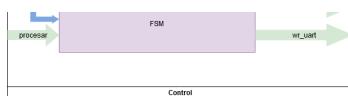


FIGURA 3.11: FSMD del módulo registro

36

## Capítulo 3. Diseño e Implementación

La máquina de estados (FSM) se ocupa de generar cada dos ciclos de reloj un pulso para poder enviar secuencialmente los caracteres detectados. A la vez que el multiplexor va seleccionando cada elemento del vector paquete[M] según el valor del contador vigente, que se incrementa cada pulso del reloj interno generado.

Finalmente se envía un carácter ASCII '1' si el elemento i-ésimo del paquete es '1' lógico y un '0' si lo recibido es un '0' lógico. Junto con el carácter se envía la señal "wr\_uart" para indicarle a la UART que ese dato debe ser guardado en una estructura de memoria llamada FIFO (del inglés, First-In,First-Out) de salida y la señal "procesado" para indicarle al módulo de detección que ya puede recibir nuevas tramas.

La máquina de estados es ilustrada en la figura 3.12. Se añadieron dos estados para generar el pulso de reloj necesario para mantener sincronizadas las tramas. Al estado de reinicio se accede cuando el contador haya recorrido todos los elementos del paquete, igualando el valor de M, la cantidad de elementos esperados.

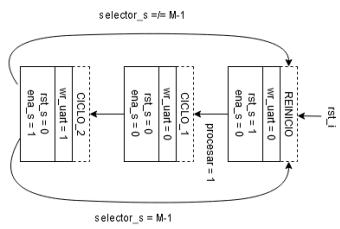


FIGURA 3.12: Estados del módulo registro

La señal "procesar" recibida de las etapas anteriores. Si la trama ingresada es incorrecta o si ya fue impresa, entonces esa señal será '0' y el registro dejará de enviar datos a la UART. En caso afirmativo ("procesar" = '1') el proceso continuará hasta que la UART indique que no pueda recibir más datos o que alguna etapa previa informe de algún error en el proceso.

## 3.7.3. Módulo selector

Para facilitar el proceso se añadió la posibilidad de elegir con uno de los switches del kit de desarrollo el puentear completamente el enclavamiento. Es decir, con una posición del switch la salida era una copia exacta de la entrada, lo cual permitió diseñar todo el proceso de detección, lectura y escritura en la UART de forma independiente al enclavamiento. Mientras que con la otra posición del switch se enviaba la señal de entrada al sistema de enclavamiento y la salida era la consecuencia de haber pasado por este proceso.

En la figura 3.13 se ilustra brevemente este módulo.

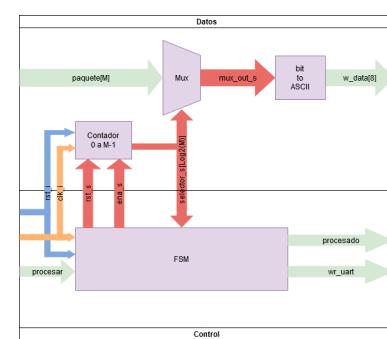


FIGURA 3.9: FSMD del módulo registro

La máquina de estados (FSM) desarrollada se ocupa de generar cada dos ciclos de reloj un pulso para poder enviar secuencialmente los caracteres detectados. A la vez que el multiplexor va seleccionando cada elemento del vector paquete[M] según el valor del contador vigente, que se incrementa cada pulso del reloj interno generado.

Finalmente se envía un carácter ASCII '1' si el elemento i-ésimo del paquete es '1' lógico y un '0' si lo recibido es un '0' lógico. Junto con el carácter se envía la señal "wr\_uart" para indicarle a la UART que ese dato debe ser guardado en una estructura de memoria llamada FIFO (del inglés, First-In,First-Out) de salida y la señal "procesado" para indicarle al módulo de detección que ya puede recibir nuevas tramas.

La máquina de estados es ilustrada en la figura 3.10.

## 3.5. Módulos de procesamiento de tramas

39

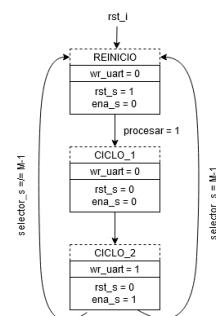


FIGURA 3.10: Estados del módulo registro

Se añadieron dos estados para generar el pulso de reloj necesario para mantener sincronizadas las tramas. Al estado de reinicio se accede cuando el contador haya recorrido todos los elementos del paquete, igualando el valor de M, la cantidad de elementos esperados.

La señal "procesar" es recibida de las etapas anteriores. Si la trama ingresada es incorrecta o si ya fue impresa, entonces esa señal será '0' y el registro dejará de enviar datos a la UART. En caso afirmativo ("procesar" = '1') el proceso continuará hasta que la UART indique que no pueda recibir más datos o que alguna etapa previa informe de algún error en el proceso.

## 3.5.3. Módulo selector

Para facilitar el proceso se añadió la posibilidad de elegir con uno de los switches del kit de desarrollo el puentear completamente el enclavamiento. En la figura 3.11 se ilustra brevemente el módulo diseñado para lograr este objetivo.

El módulo selector permite que ante un cambio en la posición del switch la salida sea una copia exacta de la entrada, lo cual permitió diseñar todo el proceso de detección, lectura y escritura en la UART de forma independiente al enclavamiento. Mientras que con la otra posición del switch se enviaba la señal de entrada al sistema de enclavamiento y la salida era la consecuencia de haber pasado por este proceso.

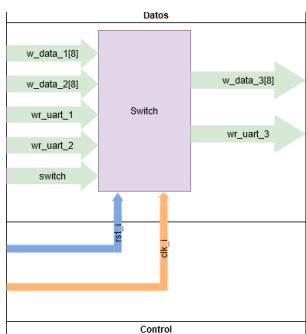


FIGURA 3.13: Diagrama de estados finitos digitales del módulo selector

La implementación es por demás sencilla: un selector que decide enviar la entrada a una salida u otra según la posición del switch, de forma asíncrona. Aunque no solo envía el dato sino también la ráfaga de pulsos asociada para su correcta escritura en la UART.

### 3.8. Módulo de comunicación UART

En el diseño de los módulos para la interfaz UART con el exterior se utilizó un modelo básico aportado por los docentes, pero fueron necesarias varias premisas para modificarlo y poder automatizarlo:

- Se deben tener dos FIFOs distintas, una de entrada y la otra de salida.
- El tamaño de las FIFOs debe adaptarse a la topología: redes más grandes necesitarán FIFOs más grandes y redes más pequeñas requerirán FIFOs más pequeñas.
- Ambas FIFOs no pueden tener tamaño idéntico.
- Se deben incluir señales que indiquen al sistema si se tienen nuevos datos del exterior o si es posible recibir nuevos datos procesados para su posterior impresión.
- Cada cierto tiempo ambas FIFOs deberán vaciarse en su totalidad.

Se presenta en la figura 3.14 un diagrama en bloques de la UART.

Los bloques de recepción y transmisión son funcionalmente idénticos, pero instanciados de forma diferente para que adopten tamaños distintos y sean conectados a señales distintas, ya que su rol no es el mismo. La FIFO de entrada es la encargada de almacenar los valores ingresados en la plataforma con un baud-rate

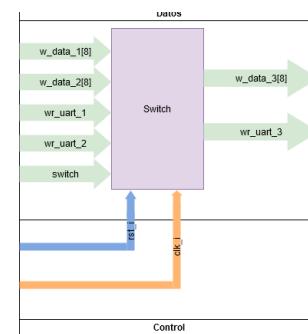


FIGURA 3.11: Diagrama de estados finitos digitales del módulo selector

La implementación permite enviar la entrada a una salida u otra según la posición del switch, de forma asíncrona. Aunque no solo envía el dato sino también la ráfaga de pulsos asociada para su correcta escritura en la UART.

### 3.6. Módulo de comunicación UART

En el diseño de los módulos para la interfaz UART con el exterior se utilizó un modelo aportado por los docentes, pero fueron necesarias varias premisas para modificarlo y poder automatizarlo:

- Se deben tener dos FIFOs distintas, una de entrada y la otra de salida.
- El tamaño de las FIFOs debe adaptarse a la topología: redes más grandes necesitarán FIFOs más grandes y redes más pequeñas requerirán FIFOs más pequeñas.
- Ambas FIFOs no pueden tener tamaño idéntico.
- Se deben incluir señales que indiquen al sistema si se tienen nuevos datos del exterior o si es posible recibir nuevos datos procesados para su posterior impresión.
- Cada cierto tiempo ambas FIFOs deberán vaciarse en su totalidad.

Se presenta en la figura 3.12 un diagrama en bloques de la UART.

Los bloques de recepción y transmisión son funcionalmente idénticos, pero instanciados de forma diferente para que adopten tamaños distintos y sean conectados a señales distintas, ya que su rol no es el mismo. La FIFO de entrada es la encargada de almacenar los valores ingresados (conector amarillo) en la plataforma con un baud-rate definido y envía su contenido al sistema junto con una serie

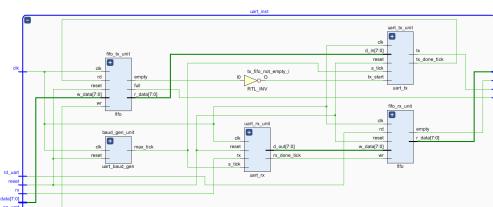


FIGURA 3.14: Diagrama en bloques de la UART

definido y envía su contenido al sistema junto con una serie de pulsos para indicar cuándo deben ser leídos. La FIFO de salida, en cambio, debe almacenar los resultados del sistema según una serie de pulsos generada por el mismo sistema, para luego imprimir el resultado por el puerto serie con el baud-rate definido.

La FIFO de entrada se utiliza para almacenar una trama de la forma '<Mensaje de largo N>' por lo que espera al menos N+2 bytes, mientras que la FIFO de salida espera mensajes de la forma 'Mensaje de largo M'. Ya que N > M es claro que N + 2 > M en la mayoría de los casos. Aunque también puede ocurrir que la diferencia entre ambos no sea tanta y al implementar el tamaño de las FIFOs en potencias de 2 terminen ambas FIFOs con el mismo tamaño.

Con este criterio de diseño, en todos los demás casos, la FIFO de salida deberá ser 50 % menor que la de entrada, lo que representa un ahorro de 25 % de los recursos estimados.

### 3.9. Interfaz de comunicación Python

El algoritmo analizador de redes finaliza ejecutando la conexión de la consola utilizada con la plataforma FPGA. Presenta al usuario un menú de opciones para ocupar/desocupar ciertos tramos de la vía, cambiar el aspecto de ciertos semáforos, pedir que una barrera suba o baje o incluso modificar la posición de una máquina de cambios.

En la figura 3.15 se ilustra el menú diseñado para ingresar los comandos al sistema. Ya que existe una interfaz mucho más avanzada desarrollada por otra parte del equipo de investigación en UTN Haedo, no se ha invertido demasiado tiempo en tener una interfaz propia para realizar pruebas a mayor escala.

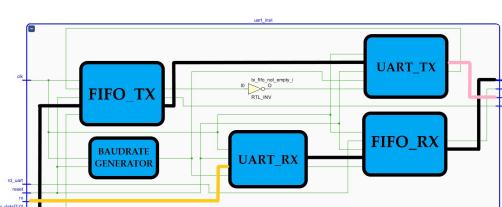


FIGURA 3.12: Diagrama en bloques de la UART

de pulsos para indicar cuándo deben ser leídos. La FIFO de salida, en cambio, debe almacenar los resultados del sistema según una serie de pulsos generada por el mismo sistema, para luego imprimir el resultado por el puerto serie (conector rosa) con el baud-rate definido.

La FIFO de entrada se utiliza para almacenar una trama de la forma '<Mensaje de largo N>' (conector negro) por lo que espera al menos N+2 bytes, mientras que la FIFO de salida espera mensajes de la forma 'Mensaje de largo M' (conector negro). Ya que N > M es claro que N + 2 > M en la mayoría de los casos. Aunque también puede ocurrir que la diferencia entre ambos no sea tanta y al implementar el tamaño de las FIFOs en potencias de 2 terminen ambas FIFOs con el mismo tamaño.

Con este criterio de diseño, en todos los demás casos, la FIFO de salida tendrá el mismo tamaño que la FIFO de entrada o a lo sumo será 50 % menor, lo que representa un ahorro de 25 % de los recursos estimados. Por ejemplo, si necesito la entrada tiene 15 elementos y la salida 7 elementos y la asigno el mismo tamaño a ambas FIFOs; tanto la FIFO de entrada como la de salida necesitarán 16 bits cada una, dando un total de 32 bits. Pero si se aplica el criterio de tamaños desacoplados, entonces para la FIFO de salida podrían asignarse solamente 8 bits, dando un total de 24 bits, un 25 % menos que los 32 bits que necesitaría si ambas FIFOs quedaran definidas según los datos de la entrada.

### 3.7. Interfaz de comunicación Python

El algoritmo analizador de redes finaliza ejecutando la conexión de la consola utilizada con la plataforma FPGA. Presenta al usuario un menú de opciones para ocupar/desocupar ciertos tramos de la vía, cambiar el aspecto de ciertos semáforos, pedir que una barrera suba o baje o incluso modificar la posición de una máquina de cambios.

En la figura 3.13 se ilustra el menú diseñado para ingresar los comandos al sistema. Ya que existe una interfaz mucho más avanzada desarrollada por otra parte del equipo de investigación en UTN Haedo, no se ha invertido demasiado tiempo en tener una interfaz propia para realizar pruebas a mayor escala.

```

conexión al puerto serie .....
Ingrese el puerto serie, ejemplos: /dev/ttyUSB0 , COM1
O bien ingrese 'l' para /dev/ttyUSB0, o 'w' para COM9
W
COM9 abierto.

Comandos disponibles .....
'h' (help) Imprime esta lista de comandos.
'q' (quit) Salir del programa.
'1' Insertar tren
'2' Remover tren
'3' Modificar aspecto de semáforo
'4' Modificar posición de la máquina de cambios
'>' Avanzar todos los trenes
'<' Retroceder todos los trenes

```

FIGURA 3.15: Menú de opciones para comunicarse con el sistema.

Todos los cambios repercuten en el grafo mostrado en pantalla, que cambiará los colores de los semáforos y de los nodos para representar la ocupación del tren; así como también el color de las aristas para indicar la posición de los cambios.

```

conexión al puerto serie .....
Ingrese el puerto serie, ejemplos: /dev/ttyUSB0 , COM1
O bien ingrese 'l' para /dev/ttyUSB0, o 'w' para COM9
W
COM9 abierto.

Comandos disponibles .....
'h' (help) Imprime esta lista de comandos.
'q' (quit) Salir del programa.
'1' Insertar tren
'2' Remover tren
'3' Modificar aspecto de semáforo
'4' Modificar posición de la máquina de cambios
'>' Avanzar todos los trenes
'<' Retroceder todos los trenes

```

FIGURA 3.13: Menú de opciones para comunicarse con el sistema.

Todos los cambios repercuten en el grafo mostrado en pantalla (figura 3.14), que cambiará los colores de los semáforos y de los nodos para representar la ocupación del tren; así como también el color de las aristas para indicar la posición de los cambios.

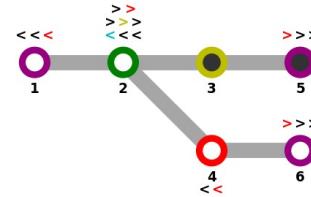


FIGURA 3.14: Grafo modificado en tiempo real por la FPGA.

## Capítulo 4

### Ensayos y Resultados

#### 4.1. Validación de grafos

Se generaron varias topologías distintas, entre ellas: estaciones simples, estaciones complejas, bypass, playa de maniobras y aleatorias. Para todas ellas se generó automáticamente una tabla de enclavamientos, las cuales fueron comparadas con éxito con las tablas de enclavamientos realizadas de la forma tradicional por el Ingeniero Ramiro Ghignone de la UTN Haedo. Quien fue además el encargado de diseñar e implementar el simulador del sistema de enclavamiento, por lo que se valoró su amplia experiencia en el tema para comprobar que por dos caminos de diseño distintos se llegaban a tablas de enclavamientos idénticas.

#### 4.2. Validación del nodo

La generación automática de código instancia diferentes tipos de nodos con mas o menos salidas. A la hora de realizar los ensayos se utilizó como bloque de prueba el nodo mas completo: el nodo perteneciente al tramo de vía que incluye la máquina de cambios. De esta forma se tenía la mayor cantidad de semáforos, interacciones, vecinos y comportamientos posibles.

##### 4.2.1. Testbench del módulo nodo

Se generó el Algoritmo 4.1 para evaluar todas las posibles combinaciones de estados, definidas en la Tabla 4.1.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 --Declare the package
5 use work.my_package.all;
6
7 entity tb_nodo is
8 end tb_nodo;
9
10 architecture tb of tb_nodo is
11
12 component nodo_1 is
13 generic(
14   N : natural := 21;
15   N_SEM : natural := 7;
16   NMDC : natural := 1;
17   N_CVS : natural := 6
18 );
19
20 port(
21   Clock : in std_logic;
22   Reset : in std_logic;
23   Estado_i : in std_logic;
24   Estado_post : in std_logic;
25   Semaforo_propio_i_1 : in sem_type;
26   Semaforo_propio_o_1 : out sem_type;
27   Semaforo_cercano : out sem_type;
28   Semaforo_lejano : out sem_type;
29   Estado_o : out std_logic
30 );
31 end component;
32
33 Signal Clock : std_logic;
34 Signal Reset : std_logic;
35 Signal Estado_i : std_logic;
36 Signal Estado_post : std_logic;
37 Signal Semaforo_propio_i_1 : sem_type;
38 Signal Semaforo_propio_o_1 : sem_type;
39 Signal Semaforo_cercano : sem_type;
40 Signal Semaforo_lejano : sem_type;
41 Signal Estado_o : std_logic;
42
43 constant TbPeriod : time := 8 ns; — EDIT Put right period here
44 signal TbClock : std_logic := '0';
45 signal TbSimEnded : std_logic := '0';
46
47 begin
48
49 dut : nodo_1
50 port map (
51   Clock      => Clock,
52   Reset      => Reset,
53   Estado_i   => Estado_i,
54   Estado_post => Estado_post,
55   Semaforo_propio_i_1 => Semaforo_propio_i_1,
56   Semaforo_propio_o_1 => Semaforo_propio_o_1,
57   Semaforo_cercano => Semaforo_cercano,
```

TABLA 4.1: Combinaciones posibles

Tramo propia	Tramo vecino	Aspecto vecino
Ocupado	Ocupado	Rojo
Ocupado	Ocupado	Amarillo
Ocupado	Ocupado	Verde
Ocupado	Libre	Rojo
Ocupado	Libre	Amarillo
Ocupado	Libre	Verde
Libre	Ocupado	Rojo
Libre	Ocupado	Amarillo
Libre	Ocupado	Verde
Libre	Libre	Rojo
Libre	Libre	Amarillo
Libre	Libre	Verde

## Capítulo 4

### Ensayos y Resultados

#### 4.1. Validación de grafos

Se generaron varias topologías distintas, entre ellas: estaciones simples, estaciones complejas, bypass, playa de maniobras y aleatorias. Para todas ellas se generó automáticamente una tabla de enclavamientos, las cuales fueron comparadas con éxito con las tablas de enclavamientos realizadas de la forma tradicional por el Ingeniero Ramiro Ghignone de la UTN Haedo. Quien fue además el encargado de diseñar e implementar el simulador del sistema de enclavamiento, por lo que se valoró su amplia experiencia en el tema para comprobar que por dos caminos de diseño distintos se llegaban a tablas de enclavamientos idénticas.

#### 4.2. Validación del nodo

La generación automática de código instancia diferentes tipos de nodos con mas o menos salidas. A la hora de realizar los ensayos se utilizó como bloque de prueba el nodo mas completo: el nodo perteneciente al tramo de vía que incluye la máquina de cambios. De esta forma se tenía la mayor cantidad de semáforos, interacciones, vecinos y comportamientos posibles.

##### 4.2.1. Testbench del módulo nodo

Se generó el Algoritmo 4.1 para evaluar todas las posibles combinaciones de estados, definidas en la Tabla 4.1.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 --Declare the package
5 use work.my_package.all;
6
7 entity tb_nodo is
8 end tb_nodo;
9
10 architecture tb of tb_nodo is
11
12 component nodo_1 is
13 generic(
14   N : natural := 21;
15   N_SEM : natural := 7;
16   NMDC : natural := 1;
17   N_CVS : natural := 6
18 );
19
20 port(
21   Clock : in std_logic;
22   Reset : in std_logic;
23   Estado_i : in std_logic;
24   Estado_post : in std_logic;
25   Semaforo_propio_i_1 : in sem_type;
26   Semaforo_propio_o_1 : out sem_type;
27   Semaforo_cercano : out sem_type;
28   Semaforo_lejano : out sem_type;
29   Estado_o : out std_logic
30 );
31 end component;
32
33 Signal Clock : std_logic;
34 Signal Reset : std_logic;
35 Signal Estado_i : std_logic;
36 Signal Estado_post : std_logic;
37 Signal Semaforo_propio_i_1 : sem_type;
38 Signal Semaforo_propio_o_1 : sem_type;
39 Signal Semaforo_cercano : sem_type;
40 Signal Semaforo_lejano : sem_type;
41 Signal Estado_o : std_logic;
42
43 constant TbPeriod : time := 8 ns; — EDIT Put right period here
44 signal TbClock : std_logic := '0';
45 signal TbSimEnded : std_logic := '0';
46
47 begin
48
49 dut : nodo_1
50 port map (
51   Clock      => Clock,
52   Reset      => Reset,
53   Estado_i   => Estado_i,
54   Estado_post => Estado_post,
55   Semaforo_propio_i_1 => Semaforo_propio_i_1,
56   Semaforo_propio_o_1 => Semaforo_propio_o_1,
57   Semaforo_cercano => Semaforo_cercano,
```

TABLA 4.1: Combinaciones posibles

Tramo propia	Tramo vecino	Aspecto vecino
Ocupado	Ocupado	Rojo
Ocupado	Ocupado	Amarillo
Ocupado	Ocupado	Verde
Ocupado	Libre	Rojo
Ocupado	Libre	Amarillo
Ocupado	Libre	Verde
Libre	Ocupado	Rojo
Libre	Ocupado	Amarillo
Libre	Ocupado	Verde
Libre	Libre	Rojo
Libre	Libre	Amarillo
Libre	Libre	Verde

```

20 port(
21   Clock : in std_logic;
22   Reset : in std_logic;
23   Estado_i : in std_logic;
24   Estado_post : in std_logic;
25   Semaforo_propio_i_1 : in sem_type;
26   Semaforo_propio_o_1 : out sem_type;
27   Semaforo_cercano : out sem_type;
28   Semaforo_lejano : out sem_type;
29   Estado_o : out std_logic
30 );
31 end component;
32
33 Signal Clock : std_logic;
34 Signal Reset : std_logic;
35 Signal Estado_i : std_logic;
36 Signal Estado_post : std_logic;
37 Signal Semaforo_propio_i_1 : sem_type;
38 Signal Semaforo_propio_o_1 : sem_type;
39 Signal Semaforo_cercano : sem_type;
40 Signal Semaforo_lejano : sem_type;
41 Signal Estado_o : std_logic;
42
43 constant TbPeriod : time := 8 ns; — EDIT Put right period here
44 signal TbClock : std_logic := '0';
45 signal TbSimEnded : std_logic := '0';
46
47 begin
48
49 dut : nodo_1
50 port map (
51   Clock      => Clock,
52   Reset      => Reset,
53   Estado_i   => Estado_i,
54   Estado_post => Estado_post,
55   Semaforo_propio_i_1 => Semaforo_propio_i_1,
56   Semaforo_propio_o_1 => Semaforo_propio_o_1,
57   Semaforo_cercano => Semaforo_cercano,
```

## 4.2. Validación del nodo

43

```

58     Semaforo_lejano => Semaforo_lejano ,
59     Estado_o => Estado_o
60   );
61
62 — Clock generation
63 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
64   '0';
65 — EDIT: Check that clk_i is really your main clock signal
66 Clock <= TbClock;
67
68 datos : process
69 begin
70   Reset <= '0';
71   wait for 1 * TbPeriod;
72   Reset <= '1';
73
74   — (0,0,0,0):
75   Estado_i <= '0';
76   Estado_post <= '0';
77   Semaforo_propio_i_1.lsb <= '0';
78   Semaforo_propio_i_1.msb <= '0';
79   wait for 20 * TbPeriod;
80   — (0,0,0,1):
81   Estado_i <= '0';
82   Estado_post <= '0';
83   Semaforo_propio_i_1.lsb <= '0';
84   Semaforo_propio_i_1.msb <= '1';
85   wait for 20 * TbPeriod;
86   — (0,0,1,1):
87   Estado_i <= '0';
88   Estado_post <= '0';
89   Semaforo_propio_i_1.lsb <= '1';
90   Semaforo_propio_i_1.msb <= '1';
91   wait for 20 * TbPeriod;
92   — (0,1,0,0):
93   Estado_i <= '0';
94   Estado_post <= '1';
95   Semaforo_propio_i_1.lsb <= '0';
96   Semaforo_propio_i_1.msb <= '0';
97   wait for 20 * TbPeriod;
98   — (0,1,0,1):
99   Estado_i <= '0';
100  Estado_post <= '1';
101  Semaforo_propio_i_1.lsb <= '0';
102  Semaforo_propio_i_1.msb <= '1';
103  wait for 20 * TbPeriod;
104  — (0,1,1,1):
105  Estado_i <= '0';
106  Estado_post <= '1';
107  Semaforo_propio_i_1.lsb <= '1';
108  Semaforo_propio_i_1.msb <= '1';
109  wait for 20 * TbPeriod;
110  — (1,0,0,0):
111  Estado_i <= '1';
112  Estado_post <= '0';
113  Semaforo_propio_i_1.lsb <= '0';
114  Semaforo_propio_i_1.msb <= '0';
115  wait for 20 * TbPeriod;
116  — (1,0,0,1):
117  Estado_i <= '1';
118  Estado_post <= '0';
119  Semaforo_propio_i_1.lsb <= '0';

```

44

## Capítulo 4. Ensayos y Resultados

```

120 Semaforo_propio_i_1.msb <= '1';
121 wait for 20 * TbPeriod;
122 — (1,0,1,1):
123 Estado_i <= '1';
124 Estado_post <= '0';
125 Semaforo_propio_i_1.lsb <= '1';
126 Semaforo_propio_i_1.msb <= '1';
127 wait for 20 * TbPeriod;
128 — (1,1,0,0):
129 Estado_i <= '1';
130 Estado_post <= '1';
131 Semaforo_propio_i_1.lsb <= '0';
132 Semaforo_propio_i_1.msb <= '0';
133 wait for 20 * TbPeriod;
134 — (1,1,0,1):
135 Estado_i <= '1';
136 Estado_post <= '1';
137 Semaforo_propio_i_1.lsb <= '0';
138 Semaforo_propio_i_1.msb <= '1';
139 wait for 20 * TbPeriod;
140 — (1,1,1,1):
141 Estado_i <= '1';
142 Estado_post <= '1';
143 Semaforo_propio_i_1.lsb <= '1';
144 Semaforo_propio_i_1.msb <= '1';
145 wait for 20 * TbPeriod;
146
147 wait for 500 * TbPeriod;
148 —TbSimEnded <= '1';
149 end process;
150
151 end tb;
152
153 — Configuration block below is required by some simulators. Usually no
154 need to edit.
155 configuration cfg_tb_nodo of tb_nodo is
156   for tb
157   end for;
158 end cfg_tb_nodo;

```

ALGORITMO 4.1: Testbench del módulo nodo

## 4.2.2. Resultados obtenidos

En la figura 4.1 se muestra el resultado obtenido. Siempre que el tramo está ocupado el semáforo cambia su aspecto a rojo. Si el tramo analizado está desocupado se desprenden varios casos:

- Si el vecino está ocupado: el semáforo analizado estará en aspecto rojo.
- Si el vecino está desocupado:
  - Si el semáforo vecino está en rojo: el semáforo analizado estará en aspecto amarillo.
  - Si el semáforo vecino está en amarillo: el semáforo analizado estará en aspecto verde.

## 4.2. Validación del nodo

45

```

58     Semaforo_lejano => Semaforo_lejano ,
59     Estado_o => Estado_o
60   );
61
62 — Clock generation
63 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
64   '0';
65 — EDIT: Check that clk_i is really your main clock signal
66 Clock <= TbClock;
67
68 datos : process
69 begin
70   Reset <= '0';
71   wait for 1 * TbPeriod;
72   Reset <= '1';
73
74   — (0,0,0,0):
75   Estado_i <= '0';
76   Estado_post <= '0';
77   Semaforo_propio_i_1.lsb <= '0';
78   Semaforo_propio_i_1.msb <= '0';
79   wait for 20 * TbPeriod;
80   — (0,0,0,1):
81   Estado_i <= '0';
82   Estado_post <= '0';
83   Semaforo_propio_i_1.lsb <= '0';
84   Semaforo_propio_i_1.msb <= '1';
85   wait for 20 * TbPeriod;
86   — (0,0,1,1):
87   Estado_i <= '0';
88   Estado_post <= '0';
89   Semaforo_propio_i_1.lsb <= '1';
90   Semaforo_propio_i_1.msb <= '1';
91   wait for 20 * TbPeriod;
92   — (0,1,0,0):
93   Estado_i <= '0';
94   Estado_post <= '1';
95   Semaforo_propio_i_1.lsb <= '0';
96   Semaforo_propio_i_1.msb <= '0';
97   wait for 20 * TbPeriod;
98   — (0,1,0,1):
99   Estado_i <= '0';
100  Estado_post <= '1';
101  Semaforo_propio_i_1.lsb <= '0';
102  Semaforo_propio_i_1.msb <= '1';
103  wait for 20 * TbPeriod;
104  — (0,1,1,1):
105  Estado_i <= '0';
106  Estado_post <= '1';
107  Semaforo_propio_i_1.lsb <= '1';
108  Semaforo_propio_i_1.msb <= '1';
109  wait for 20 * TbPeriod;
110  — (1,0,0,0):
111  Estado_i <= '0';
112  Estado_post <= '1';
113  Semaforo_propio_i_1.lsb <= '0';
114  Semaforo_propio_i_1.msb <= '0';
115  wait for 20 * TbPeriod;
116  — (1,0,0,1):
117  Estado_i <= '1';
118  Estado_post <= '0';
119  Semaforo_propio_i_1.lsb <= '0';

```

46

## Capítulo 4. Ensayos y Resultados

```

120 Semaforo_propio_i_1.msb <= '1';
121 wait for 20 * TbPeriod;
122 — (1,0,1,1):
123 Estado_i <= '1';
124 Estado_post <= '0';
125 Semaforo_propio_i_1.lsb <= '1';
126 Semaforo_propio_i_1.msb <= '1';
127 wait for 20 * TbPeriod;
128 — (1,1,0,0):
129 Estado_i <= '1';
130 Estado_post <= '1';
131 Semaforo_propio_i_1.lsb <= '0';
132 Semaforo_propio_i_1.msb <= '0';
133 wait for 20 * TbPeriod;
134 — (1,1,0,1):
135 Estado_i <= '1';
136 Estado_post <= '1';
137 Semaforo_propio_i_1.lsb <= '0';
138 Semaforo_propio_i_1.msb <= '1';
139 wait for 20 * TbPeriod;
140 — (1,1,1,1):
141 Estado_i <= '1';
142 Estado_post <= '1';
143 Semaforo_propio_i_1.lsb <= '1';
144 Semaforo_propio_i_1.msb <= '1';
145 wait for 20 * TbPeriod;
146
147 wait for 500 * TbPeriod;
148 —TbSimEnded <= '1';
149 end process;
150
151 end tb;
152
153 — Configuration block below is required by some simulators. Usually no
154 need to edit.
155 configuration cfg_tb_nodo of tb_nodo is
156   for tb
157   end for;
158 end cfg_tb_nodo;

```

ALGORITMO 4.1: Testbench del módulo nodo

## 4.2.2. Resultados obtenidos

En la figura 4.1 se muestra el resultado obtenido. Siempre que el tramo está ocupado el semáforo cambia su aspecto a rojo. Si el tramo analizado está desocupado se desprenden varios casos:

- Si el vecino está ocupado: el semáforo analizado estará en aspecto rojo.
- Si el vecino está desocupado:
  - Si el semáforo vecino está en rojo: el semáforo analizado estará en aspecto amarillo.
  - Si el semáforo vecino está en amarillo: el semáforo analizado estará en aspecto verde.

## 4.3. Validación de la máquina de cambios

45



FIGURA 4.1: Simulación de un nodo.

El haber validado el nodo genérico que incluye todos los posibles estados que los nodos menos complejos heredarán, se consideró que el ensayo fue un éxito.

## 4.3. Validación de la máquina de cambios

El módulo de la máquina de cambios tiene como función el conectar al estado anterior con el estado posterior o al estado anterior con el estado desvío según la posición del cambio de vías, como se indica en la Tabla 4.2.

TABLA 4.2: Combinaciones posibles

Posición del cambio	Estado anterior	Estado posterior	Estado desvío
Normal	Ocupado	Ocupado	Ocupado
Normal	Ocupado	Ocupado	Libre
Normal	Ocupado	Libre	Ocupado
Normal	Ocupado	Libre	Libre
Normal	Libre	Ocupado	Ocupado
Normal	Libre	Ocupado	Libre
Normal	Libre	Libre	Ocupado
Inversa	Ocupado	Ocupado	Ocupado
Inversa	Ocupado	Ocupado	Libre
Inversa	Ocupado	Libre	Ocupado
Inversa	Ocupado	Libre	Libre
Inversa	Libre	Ocupado	Ocupado
Inversa	Libre	Ocupado	Libre
Inversa	Libre	Libre	Ocupado
Inversa	Libre	Libre	Libre

## 4.3.1. Testbench del módulo de la máquina de cambios

Se generó el Algoritmo 4.2 donde se ensayan todas las combinaciones definidas en la Tabla 4.2.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 --Declare the package
5 use work.my_package.all;
6
7

```

## 4.3. Validación de la máquina de cambios

47

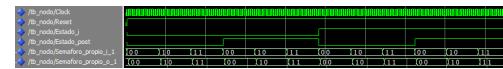


FIGURA 4.1: Simulación de un nodo.

El haber validado el nodo genérico que incluye todos los posibles estados que los nodos menos complejos heredarán, se consideró que el ensayo fue un éxito.

## 4.3. Validación de la máquina de cambios

El módulo de la máquina de cambios tiene como función el conectar al estado anterior con el estado posterior o al estado anterior con el estado desvío según la posición del cambio de vías, como se indica en la Tabla 4.2.

TABLA 4.2: Combinaciones posibles

Posición del cambio	Estado anterior	Estado posterior	Estado desvío
Normal	Ocupado	Ocupado	Ocupado
Normal	Ocupado	Ocupado	Libre
Normal	Ocupado	Libre	Ocupado
Normal	Ocupado	Libre	Libre
Normal	Libre	Ocupado	Ocupado
Normal	Libre	Ocupado	Libre
Normal	Libre	Libre	Ocupado
Inversa	Ocupado	Ocupado	Ocupado
Inversa	Ocupado	Ocupado	Libre
Inversa	Ocupado	Libre	Ocupado
Inversa	Ocupado	Libre	Libre
Inversa	Libre	Ocupado	Ocupado
Inversa	Libre	Ocupado	Libre
Inversa	Libre	Libre	Ocupado
Inversa	Libre	Libre	Libre

## 4.3.1. Testbench del módulo de la máquina de cambios

Se generó el Algoritmo 4.2 donde se ensayan todas las combinaciones definidas en la Tabla 4.2.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 --Declare the package
5 use work.my_package.all;
6
7

```

46

Capítulo 4. Ensayos y Resultados

```

8 entity tb_cambio is
9 end tb_cambio;
10
11 architecture tb of tb_cambio is
12
13 component cambio_1 is
14 generic(
15   N : natural := 21;
16   N_SEM : natural := 7;
17   N_MDC : natural := 1;
18   N_CVS : natural := 6
19 );
20 port(
21   Clock : in std_logic;
22   Reset : in std_logic;
23   Estado_ant_i : in std_logic;
24   Estado_post_i : in std_logic;
25   Estado_desv_i : in std_logic;
26   Estado_ant_o : out std_logic;
27   Estado_post_o : out std_logic;
28   Estado_desv_o : out std_logic;
29   Cambio_i : in std_logic;
30   Cambio_o : out std_logic
31 );
32 end component;
33
34 Signal Clock : std_logic;
35 Signal Reset : std_logic;
36 Signal Estado_ant_i : std_logic;
37 Signal Estado_post_i : std_logic;
38 Signal Estado_desv_i : std_logic;
39 Signal Estado_ant_o : std_logic;
40 Signal Estado_post_o : std_logic;
41 Signal Estado_desv_o : std_logic;
42 Signal Cambio_i : std_logic;
43 Signal Cambio_o : std_logic;
44
45 constant TbPeriod : time := 8 ns; — EDIT Put right period here
46 signal TbClock : std_logic := '0';
47 signal TbSimEnded : std_logic := '0';
48
49 begin
50
51   dut : cambio_1
52   port map (
53     Clock      => Clock,
54     Reset      => Reset,
55     Estado_ant_i => Estado_ant_i,
56     Estado_post_i => Estado_post_i,
57     Estado_desv_i => Estado_desv_i,
58     Estado_ant_o => Estado_ant_o,
59     Estado_post_o => Estado_post_o,
60     Estado_desv_o => Estado_desv_o,
61     Cambio_i    => Cambio_i,
62     Cambio_o    => Cambio_o
63   );
64
65   — Clock generation
66   TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
67   '0';
68
69   — EDIT: Check that clk i is really your main clock signal

```

48

Capítulo 4. Ensayos y Resultados

```

8 entity tb_cambio is
9 end tb_cambio;
10
11 architecture tb of tb_cambio is
12
13 component cambio_1 is
14 generic(
15   N : natural := 21;
16   N_SEM : natural := 7;
17   N_MDC : natural := 1;
18   N_CVS : natural := 6
19 );
20 port(
21   Clock : in std_logic;
22   Reset : in std_logic;
23   Estado_ant_i : in std_logic;
24   Estado_post_i : in std_logic;
25   Estado_desv_i : in std_logic;
26   Estado_ant_o : out std_logic;
27   Estado_post_o : out std_logic;
28   Estado_desv_o : out std_logic;
29   Cambio_i : in std_logic;
30   Cambio_o : out std_logic
31 );
32 end component;
33
34 Signal Clock : std_logic;
35 Signal Reset : std_logic;
36 Signal Estado_ant_i : std_logic;
37 Signal Estado_post_i : std_logic;
38 Signal Estado_desv_i : std_logic;
39 Signal Estado_ant_o : std_logic;
40 Signal Estado_post_o : std_logic;
41 Signal Estado_desv_o : std_logic;
42 Signal Cambio_i : std_logic;
43 Signal Cambio_o : std_logic;
44
45 constant TbPeriod : time := 8 ns; — EDIT Put right period here
46 signal TbClock : std_logic := '0';
47 signal TbSimEnded : std_logic := '0';
48
49 begin
50
51   dut : cambio_1
52   port map (
53     Clock      => Clock,
54     Reset      => Reset,
55     Estado_ant_i => Estado_ant_i,
56     Estado_post_i => Estado_post_i,
57     Estado_desv_i => Estado_desv_i,
58     Estado_ant_o => Estado_ant_o,
59     Estado_post_o => Estado_post_o,
60     Estado_desv_o => Estado_desv_o,
61     Cambio_i    => Cambio_i,
62     Cambio_o    => Cambio_o
63   );
64
65   — Clock generation
66   TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
67   '0';
68
69   — EDIT: Check that clk i is really your main clock signal

```

```
69    Clock <= TbClock;
```

## 4.3. Validación de la máquina de cambios

47

```
70
71  datos : process
72  begin
73      Reset <= '0';
74      wait for 1 * TbPeriod;
75      Reset <= '1';
76
77      — (0,0,0,0):
78      Cambio_i <= '0';
79      Estado_ante_i <= '0';
80      Estado_post_i <= '0';
81      Estado_desv_i <= '0';
82      wait for 20 * TbPeriod;
83      — (0,0,0,1):
84      Cambio_i <= '0';
85      Estado_ante_i <= '0';
86      Estado_post_i <= '0';
87      Estado_desv_i <= '1';
88      wait for 20 * TbPeriod;
89      — (0,0,1,0):
90      Cambio_i <= '0';
91      Estado_ante_i <= '0';
92      Estado_post_i <= '1';
93      Estado_desv_i <= '0';
94      wait for 20 * TbPeriod;
95      — (0,0,1,1):
96      Cambio_i <= '0';
97      Estado_ante_i <= '0';
98      Estado_post_i <= '1';
99      Estado_desv_i <= '1';
100     wait for 20 * TbPeriod;
101    — (0,1,0,0):
102    Cambio_i <= '0';
103    Estado_ante_i <= '1';
104    Estado_post_i <= '0';
105    Estado_desv_i <= '0';
106    wait for 20 * TbPeriod;
107    — (0,1,0,1):
108    Cambio_i <= '0';
109    Estado_ante_i <= '1';
110    Estado_post_i <= '0';
111    Estado_desv_i <= '1';
112    wait for 20 * TbPeriod;
113    — (0,1,1,0):
114    Cambio_i <= '0';
115    Estado_ante_i <= '1';
116    Estado_post_i <= '1';
117    Estado_desv_i <= '0';
118    wait for 20 * TbPeriod;
119    — (0,1,1,1):
120    Cambio_i <= '0';
121    Estado_ante_i <= '1';
122    Estado_post_i <= '1';
123    Estado_desv_i <= '1';
124    wait for 20 * TbPeriod;
125    — (1,0,0,0):
126    Cambio_i <= '1';
127    Estado_ante_i <= '0';
128    Estado_post_i <= '0';
129    Estado_desv_i <= '0';
130    wait for 20 * TbPeriod;
131    — (1,0,0,1):
132    Cambio_i <= '1';
```

## 4.3. Validación de la máquina de cambios

49

```
70
71  datos : process
72  begin
73      Reset <= '0';
74      wait for 1 * TbPeriod;
75      Reset <= '1';
76
77      — (0,0,0,0):
78      Cambio_i <= '0';
79      Estado_ante_i <= '0';
80      Estado_post_i <= '0';
81      Estado_desv_i <= '0';
82      wait for 20 * TbPeriod;
83      — (0,0,0,1):
84      Cambio_i <= '0';
85      Estado_ante_i <= '0';
86      Estado_post_i <= '0';
87      Estado_desv_i <= '1';
88      wait for 20 * TbPeriod;
89      — (0,0,1,0):
90      Cambio_i <= '0';
91      Estado_ante_i <= '0';
92      Estado_post_i <= '1';
93      Estado_desv_i <= '0';
94      wait for 20 * TbPeriod;
95      — (0,0,1,1):
96      Cambio_i <= '0';
97      Estado_ante_i <= '0';
98      Estado_post_i <= '1';
99      Estado_desv_i <= '1';
100     wait for 20 * TbPeriod;
101    — (0,1,0,0):
102    Cambio_i <= '0';
103    Estado_ante_i <= '1';
104    Estado_post_i <= '0';
105    Estado_desv_i <= '0';
106    wait for 20 * TbPeriod;
107    — (0,1,0,1):
108    Cambio_i <= '0';
109    Estado_ante_i <= '1';
110    Estado_post_i <= '0';
111    Estado_desv_i <= '1';
112    wait for 20 * TbPeriod;
113    — (0,1,1,0):
114    Cambio_i <= '0';
115    Estado_ante_i <= '1';
116    Estado_post_i <= '1';
117    Estado_desv_i <= '0';
118    wait for 20 * TbPeriod;
119    — (0,1,1,1):
120    Cambio_i <= '0';
121    Estado_ante_i <= '1';
122    Estado_post_i <= '1';
123    Estado_desv_i <= '1';
124    wait for 20 * TbPeriod;
125    — (1,0,0,0):
126    Cambio_i <= '1';
127    Estado_ante_i <= '0';
128    Estado_post_i <= '0';
129    Estado_desv_i <= '0';
130    wait for 20 * TbPeriod;
131    — (1,0,0,1):
132    Cambio_i <= '1';
```

48

## Capítulo 4. Ensayos y Resultados

```
133  Estado_ante_i <= '0';
134  Estado_post_i <= '0';
135  Estado_desv_i <= '1';
136  wait for 20 * TbPeriod;
137  — (1,0,1,0):
138  Cambio_i <= '1';
139  Estado_ante_i <= '0';
140  Estado_post_i <= '1';
141  Estado_desv_i <= '0';
142  wait for 20 * TbPeriod;
143  — (1,0,1,1):
144  Cambio_i <= '1';
145  Estado_ante_i <= '0';
146  Estado_post_i <= '1';
147  Estado_desv_i <= '1';
148  wait for 20 * TbPeriod;
149  — (1,1,0,0):
150  Cambio_i <= '1';
151  Estado_ante_i <= '1';
152  Estado_post_i <= '0';
153  Estado_desv_i <= '0';
154  wait for 20 * TbPeriod;
155  — (1,1,0,1):
156  Cambio_i <= '1';
157  Estado_ante_i <= '1';
158  Estado_post_i <= '0';
159  Estado_desv_i <= '1';
160  wait for 20 * TbPeriod;
161  — (1,1,1,0):
162  Cambio_i <= '1';
163  Estado_ante_i <= '1';
164  Estado_post_i <= '1';
165  Estado_desv_i <= '0';
166  wait for 20 * TbPeriod;
167  — (1,1,1,1):
168  Cambio_i <= '1';
169  Estado_ante_i <= '1';
170  Estado_post_i <= '1';
171  Estado_desv_i <= '1';
172  wait for 20 * TbPeriod;
173
174      wait for 500 * TbPeriod;
175  —TbSimEnded <= '1';
176  end process;
177
178 end tb;
179
180 — Configuration block below is required by some simulators. Usually no
   need to edit.
181
182 configuration cfg_tb_cambio of tb_cambio is
183     for tb
184     end for;
185 end cfg_tb_cambio;
```

ALGORITMO 4.2: Testbench del módulo de la máquina de cambios

50

## Capítulo 4. Ensayos y Resultados

```
133  Estado_ante_i <= '0';
134  Estado_post_i <= '0';
135  Estado_desv_i <= '1';
136  wait for 20 * TbPeriod;
137  — (1,0,1,0):
138  Cambio_i <= '1';
139  Estado_ante_i <= '0';
140  Estado_post_i <= '1';
141  Estado_desv_i <= '0';
142  wait for 20 * TbPeriod;
143  — (1,0,1,1):
144  Cambio_i <= '1';
145  Estado_ante_i <= '0';
146  Estado_post_i <= '1';
147  Estado_desv_i <= '1';
148  wait for 20 * TbPeriod;
149  — (1,1,0,0):
150  Cambio_i <= '1';
151  Estado_ante_i <= '1';
152  Estado_post_i <= '0';
153  Estado_desv_i <= '0';
154  wait for 20 * TbPeriod;
155  — (1,1,0,1):
156  Cambio_i <= '1';
157  Estado_ante_i <= '1';
158  Estado_post_i <= '0';
159  Estado_desv_i <= '1';
160  wait for 20 * TbPeriod;
161  — (1,1,1,0):
162  Cambio_i <= '1';
163  Estado_ante_i <= '1';
164  Estado_post_i <= '1';
165  Estado_desv_i <= '0';
166  wait for 20 * TbPeriod;
167  — (1,1,1,1):
168  Cambio_i <= '1';
169  Estado_ante_i <= '1';
170  Estado_post_i <= '1';
171  Estado_desv_i <= '1';
172  wait for 20 * TbPeriod;
173
174      wait for 500 * TbPeriod;
175  —TbSimEnded <= '1';
176  end process;
177
178 end tb;
179
180 — Configuration block below is required by some simulators. Usually no
   need to edit.
181
182 configuration cfg_tb_cambio of tb_cambio is
183     for tb
184     end for;
185 end cfg_tb_cambio;
```

ALGORITMO 4.2: Testbench del módulo de la máquina de cambios

## 4.4. Validación de la UART

49

## 4.3.2. Resultados obtenidos

En la figura 4.2 se visualizan los resultados del ensayo. Cuando la posición de la máquina de cambios es normal entonces el estado anterior y el posterior están comunicados y cada uno puede ver tanto la ocupación como los semáforos del otro. Cuando la posición de la máquina de cambios es inversa entonces los estados vinculados son el anterior y el estado del nodo perteneciente al desvío.

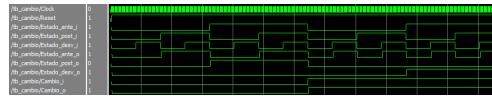


FIGURA 4.2: Simulación de la máquina de cambios.

El nodo de cambios funciona de manera sincrónica y vincula correctamente los estados de los nodos involucrados. A diferencia de los bloques de nodos donde todos los nodos implementados tienen la misma cantidad de funciones que el ensayado o menor, todos los cambios tienen las mismas funcionalidades sin ninguna limitación, por lo que este ensayo unitario es suficiente para validar su correcto funcionamiento.

## 4.4. Validación de la UART

En el diseño del sistema se contempló utilizar uno de los switches de la placa de desarrollo para poder puenteear todo el enclavamiento y probar fácilmente el funcionamiento de la UART sin el efecto del enclavamiento. En esta sección se realizó el ensayo con el switch en la posición en la cual la salida recibe el mensaje de la entrada directamente, sin pasar por ningún otro bloque.

## 4.4.1. Testbench del módulo UART

Se generó el Algoritmo 4.3 en el cual se inyecta la entrada de la UART directamente a la salida de la UART, teniendo una prueba echo. Todos los mensajes ingresados, siempre que sean válidos, pasaran a la salida para ser publicados en la terminal desde donde se originaron.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity tb_sistema is
5 end tb_sistema;
6
7 architecture tb of tb_sistema is
8
9 component sistema
10 port(
11   Clock: in std_logic;
12   Reset: in std_logic;
13   r_data: in std_logic_vector(8-1 downto 0);
14   r_disponible : in std_logic;
15   leer : out std_logic;
16   escribir : out std_logic;
17   switch1 : in std_logic;

```

## 4.4. Validación de la UART

51

## 4.3.2. Resultados obtenidos

En la figura 4.2 se visualizan los resultados del ensayo. Cuando la posición de la máquina de cambios es normal entonces el estado anterior y el posterior están comunicados y cada uno puede ver tanto la ocupación como los semáforos del otro. Cuando la posición de la máquina de cambios es inversa entonces los estados vinculados son el anterior y el estado del nodo perteneciente al desvío.



FIGURA 4.2: Simulación de la máquina de cambios.

El nodo de cambios funciona de manera sincrónica y vincula correctamente los estados de los nodos involucrados. A diferencia de los bloques de nodos donde todos los nodos implementados tienen la misma cantidad de funciones que el ensayado o menor, todos los cambios tienen las mismas funcionalidades sin ninguna limitación, por lo que este ensayo unitario es suficiente para validar su correcto funcionamiento.

## 4.4. Validación de la UART

En el diseño del sistema se contempló utilizar uno de los switches de la placa de desarrollo para poder puenteear todo el enclavamiento y probar fácilmente el funcionamiento de la UART sin el efecto del enclavamiento. En esta sección se realizó el ensayo con el switch en la posición en la cual la salida recibe el mensaje de la entrada directamente, sin pasar por ningún otro bloque.

## 4.4.1. Testbench del módulo UART

Se generó el Algoritmo 4.3 en el cual se inyecta la entrada de la UART directamente a la salida de la UART, teniendo una prueba echo. Todos los mensajes ingresados, siempre que sean válidos, pasaran a la salida para ser publicados en la terminal desde donde se originaron.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity tb_sistema is
5 end tb_sistema;
6
7 architecture tb of tb_sistema is
8
9 component sistema
10 port(
11   Clock: in std_logic;
12   Reset: in std_logic;
13   r_data: in std_logic_vector(8-1 downto 0);
14   r_disponible : in std_logic;
15   leer : out std_logic;
16   escribir : out std_logic;
17   switch1 : in std_logic;

```

50

## Capítulo 4. Ensayos y Resultados

```

18 switch2 : in std_logic;
19 reset_uart : out std_logic;
20 N : in integer;
21 leds : out std_logic_vector(4-1 downto 0);
22 led_rgb_1 : out std_logic_vector(3-1 downto 0);
23 led_rgb_2 : out std_logic_vector(3-1 downto 0);
24 w_data : out std_logic_vector(8-1 downto 0)
25 );
26 end component;
27
28
29 signal Clock : std_logic;
30 signal Reset : std_logic;
31 signal r_data : std_logic_vector (8-1 downto 0);
32 signal r_disponible : std_logic;
33 signal leer : std_logic;
34 signal escribir : std_logic;
35 signal switch1 : std_logic;
36 signal switch2 : std_logic;
37 signal reset_uart : std_logic;
38 signal N : integer;
39 signal leds : std_logic_vector(4-1 downto 0);
40 signal led_rgb_1 : std_logic_vector (3-1 downto 0);
41 signal led_rgb_2 : std_logic_vector (3-1 downto 0);
42
43 signal w_data : std_logic_vector (8-1 downto 0);
44
45 constant TbPeriod : time := 8 ns; — EDIT Put right period here
46 signal TbClock : std_logic := '0';
47 signal TbSimEnded : std_logic := '0';
48
49 constant periodo : integer := 100;
50
51 — Low-level byte-write
52 procedure Enviar_char (
53   data : in std_logic_vector(8-1 downto 0);
54   signal r_data : out std_logic_vector(8-1 downto 0);
55   signal r_disponible : out std_logic) is
56 begin
57
58   r_disponible <= '0';
59   wait for TbPeriod;
60   r_data <= data;
61   wait for TbPeriod;
62   r_disponible <= '1';
63   wait for TbPeriod;
64   r_disponible <= '0';
65
66   end Enviar_char;
67
68 begin
69
70   dut : sistema
71   port map (Clock      => Clock,
72             Reset       => Reset,
73             r_data      => r_data,

```

52

## Capítulo 4. Ensayos y Resultados

```

18 switch2 : in std_logic;
19 reset_uart : out std_logic;
20 N : in integer;
21 leds : out std_logic_vector(4-1 downto 0);
22 led_rgb_1 : out std_logic_vector(3-1 downto 0);
23 led_rgb_2 : out std_logic_vector(3-1 downto 0);
24 w_data : out std_logic_vector(8-1 downto 0)
25 );
26 end component;
27
28
29 signal Clock : std_logic;
30 signal Reset : std_logic;
31 signal r_data : std_logic_vector (8-1 downto 0);
32 signal r_disponible : std_logic;
33 signal leer : std_logic;
34 signal escribir : std_logic;
35 signal switch1 : std_logic;
36 signal switch2 : std_logic;
37 signal reset_uart : std_logic;
38 signal N : integer;
39 signal leds : std_logic_vector(4-1 downto 0);
40 signal led_rgb_1 : std_logic_vector (3-1 downto 0);
41 signal led_rgb_2 : std_logic_vector (3-1 downto 0);
42
43 signal w_data : std_logic_vector (8-1 downto 0);
44
45 constant TbPeriod : time := 8 ns; — EDIT Put right period here
46 signal TbClock : std_logic := '0';
47 signal TbSimEnded : std_logic := '0';
48
49 constant periodo : integer := 100;
50
51 — Low-level byte-write
52 procedure Enviar_char (
53   data : in std_logic_vector(8-1 downto 0);
54   signal r_data : out std_logic_vector(8-1 downto 0);
55   signal r_disponible : out std_logic) is
56 begin
57
58   r_disponible <= '0';
59   wait for TbPeriod;
60   r_data <= data;
61   wait for TbPeriod;
62   r_disponible <= '1';
63   wait for TbPeriod;
64   r_disponible <= '0';
65
66   end Enviar_char;
67
68 begin
69
70   dut : sistema
71   port map (Clock      => Clock,
72             Reset       => Reset,
73             r_data      => r_data,

```

```

74     r_disponible => r_disponible
75     leer        => leer ,
76     escribir    => escribir ,
77     switch1    => switch1 ,
78     switch2    => switch2 ,
79     reset_uart => reset_uart ,
80     N          => N,

```

#### 4.4. Validación de la UART

5

```

81     leds => leds,
82         led_rgb_1 => led_rgb_1,
83         led_rgb_2 => led_rgb_2,
84         w_data => w_data);
85
86 — Clock generation
87 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
88     '0';
89
90 — EDIT: Check that clk_i is really your main clock signal
91 Clock <= TbClock;
92
93 stimuli : process
94 begin
95     — EDIT Adapt initialization as needed
96     — r_data <= (others >= '0');
97
98     — Reset generation
99     — EDIT: Check that rst_i is really your reset signal
100    switch1 <= '1';
101    switch2 <= '0';
102        Reset <= '1';
103        wait for 100 ns;
104        Reset <= '0';
105    switch1 <= '1';
106    switch2 <= '0';
107        wait for 250000 * TbPeriod;
108
109    — Stop the clock and hence terminate the simulation
110    TbSimEnded <= '1';
111    wait;
112 end process;
113
114 data : process
115 begin
116     r_data <= "0000000";
117     r_disponible <= '0';
118
119     wait for 100 ns;
120
121 — 21 – 21 – todos 1
122 N <= 23;
123 Envir_char("00111100",r_data,r_disponible); -- <
124 Envir_char("00110001",r_data,r_disponible); -- 1
125 Envir_char("00110001",r_data,r_disponible); -- 1
126 Envir_char("00110001",r_data,r_disponible); -- 1
127 Envir_char("00110001",r_data,r_disponible); -- 1
128 Envir_char("00110001",r_data,r_disponible); -- 1
129 Envir_char("00110001",r_data,r_disponible); -- 1
130 Envir_char("00110001",r_data,r_disponible); -- 1
131 Envir_char("00110001",r_data,r_disponible); -- 1
132 Envir_char("00110001",r_data,r_disponible); -- 1
133 Envir_char("00110001",r_data,r_disponible); -- 1
134 Envir_char("00110001",r_data,r_disponible); -- 1
135 Envir_char("00110001",r_data,r_disponible); -- 1
136 Envir_char("00110001",r_data,r_disponible); -- 1
137 Envir_char("00110001",r_data,r_disponible); -- 1
138 Envir_char("00110001",r_data,r_disponible); -- 1
139 Envir_char("00110001",r_data,r_disponible); -- 1
140 Envir_char("00110001",r_data,r_disponible); -- 1
141 Envir_char("00110001",r_data,r_disponible); -- 1
142 Envir_char("00110001",r_data,r_disponible); -- 1

```

```

74      r_disponible => r_disponible
75      leer      => leer,
76      escribir   => escribir,
77      switch1    => switch1,
78      switch2    => switch2,
79      reset_uart => reset_uart,
80      N          => N,
```

#### 4.4. Validación de la UART

53

```

81    leds => leds,
82        led_rgb_1 => led_rgb_1,
83        led_rgb_2 => led_rgb_2,
84        w_data => w_data);
85
86 — Clock generation
87 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
88     '0';
89
90 — EDIT: Check that clk_i is really your main clock signal
91 Clock <= TbClock;
92
93 stimuli : process
94 begin
95     — EDIT Adapt initialization as needed
96     r_data <= (others => '0');
97
98     — Reset generation
99     — EDIT: Check that rst_i is really your reset signal
100    switch1 <= '1';
101    switch2 <= '0';
102        Reset <= '1';
103        wait for 100 ns;
104        Reset <= '0';
105    switch1 <= '1';
106    switch2 <= '0';
107        wait for 250000 * TbPeriod;
108
109    — Stop the clock and hence terminate the simulation
110    TbSimEnded <= '1';
111    wait;
112 end process;
113
114 datos : process
115 begin
116
117     r_data <= "00000000";
118     r_disponible <= '0';
119
120     wait for 100 ns;
121
122     — 21 - 21 - todos 1
123 N <= 23;
124
125 Envir_char("001111100",r_data,r_disponible); — <
126 Envir_char("00110001",r_data,r_disponible); — 1
127 Envir_char("00110001",r_data,r_disponible); — 1
128 Envir_char("00110001",r_data,r_disponible); — 1
129 Envir_char("00110001",r_data,r_disponible); — 1
130 Envir_char("00110001",r_data,r_disponible); — 1
131 Envir_char("00110001",r_data,r_disponible); — 1
132 Envir_char("00110001",r_data,r_disponible); — 1
133 Envir_char("00110001",r_data,r_disponible); — 1
134 Envir_char("00110001",r_data,r_disponible); — 1
135 Envir_char("00110001",r_data,r_disponible); — 1
136 Envir_char("00110001",r_data,r_disponible); — 1
137 Envir_char("00110001",r_data,r_disponible); — 1
138 Envir_char("00110001",r_data,r_disponible); — 1
139 Envir_char("00110001",r_data,r_disponible); — 1
140 Envir_char("00110001",r_data,r_disponible); — 1
141 Envir_char("00110001",r_data,r_disponible); — 1
142 Envir_char("00110001",r_data,r_disponible); — 1

```

```

143 Enviar_char("00110001",r_data,r_disponible); — 1
144 Enviar_char("00110001",r_data,r_disponible); — 0
145 Enviar_char("00111110",r_data,r_disponible); — >
146
147 wait for periodo * TbPeriod;
148
149 — 21 — 1
150 N <= 23;
151 Enviar_char("00111110",r_data,r_disponible); <
152 Enviar_char("00110001",r_data,r_disponible); — 0
153 Enviar_char("00110001",r_data,r_disponible); — 0
154 Enviar_char("00110001",r_data,r_disponible); — 0
155 Enviar_char("00110001",r_data,r_disponible); — 0
156 Enviar_char("00110001",r_data,r_disponible); — 0
157 Enviar_char("00110001",r_data,r_disponible); — 0
158 Enviar_char("00110001",r_data,r_disponible); — 0
159 Enviar_char("00110001",r_data,r_disponible); — 0
160 Enviar_char("00110001",r_data,r_disponible); — 0
161 Enviar_char("00110001",r_data,r_disponible); — 0
162 Enviar_char("00110001",r_data,r_disponible); — 0
163 Enviar_char("00110001",r_data,r_disponible); — 0
164 Enviar_char("00110001",r_data,r_disponible); — 0
165 Enviar_char("00110001",r_data,r_disponible); — 0
166 Enviar_char("00110001",r_data,r_disponible); — 0
167 Enviar_char("00110001",r_data,r_disponible); — 0
168 Enviar_char("00110001",r_data,r_disponible); — 0
169 Enviar_char("00110001",r_data,r_disponible); — 0
170 Enviar_char("00110001",r_data,r_disponible); — 0
171 Enviar_char("00110001",r_data,r_disponible); — 0
172 Enviar_char("00110001",r_data,r_disponible); — 1
173 Enviar_char("00111110",r_data,r_disponible); — >
174
175 wait for periodo * TbPeriod;
176
177 — 21 — 2
178 N <= 23;
179 Enviar_char("00111110",r_data,r_disponible); — <
180 Enviar_char("00110001",r_data,r_disponible); — 0
181 Enviar_char("00110001",r_data,r_disponible); — 0
182 Enviar_char("00110001",r_data,r_disponible); — 0

```

54

```

196 Enviar_char("00110000",r_data,r_disponible); — 0
197 Enviar_char("00110000",r_data,r_disponible); — 0
198 Enviar_char("00110000",r_data,r_disponible); — 0
199 Enviar_char("00110001",r_data,r_disponible); — 1
200 Enviar_char("00110000",r_data,r_disponible); — 0
201 Enviar_char("00111110",r_data,r_disponible); — >
202
203 wait for periodo * TbPeriod;
204
205 — Moviendo el "1" en todas las posiciones —

```

## 4.4. Validación de la UART

53

```

206 — Borrado para la memoria para resumir —
207
208 — Moviendo el "1" en todas las posiciones —
209
210 — 21 ~ 20
211 N <= 23;
212 Enviar_char("00111100",r_data,r_disponible); — <
213 Enviar_char("00110000",r_data,r_disponible); — 0
214 Enviar_char("00110001",r_data,r_disponible); — 1
215 Enviar_char("00110000",r_data,r_disponible); — 0
216 Enviar_char("00110000",r_data,r_disponible); — 0
217 Enviar_char("00110000",r_data,r_disponible); — 0
218 Enviar_char("00110000",r_data,r_disponible); — 0
219 Enviar_char("00110000",r_data,r_disponible); — 0
220 Enviar_char("00110000",r_data,r_disponible); — 0
221 Enviar_char("00110000",r_data,r_disponible); — 0
222 Enviar_char("00110000",r_data,r_disponible); — 0
223 Enviar_char("00110000",r_data,r_disponible); — 0
224 Enviar_char("00110000",r_data,r_disponible); — 0
225 Enviar_char("00110000",r_data,r_disponible); — 0
226 Enviar_char("00110000",r_data,r_disponible); — 0
227 Enviar_char("00110000",r_data,r_disponible); — 0
228 Enviar_char("00110000",r_data,r_disponible); — 0
229 Enviar_char("00110000",r_data,r_disponible); — 0
230 Enviar_char("00110000",r_data,r_disponible); — 0
231 Enviar_char("00110000",r_data,r_disponible); — 0
232 Enviar_char("00110000",r_data,r_disponible); — 0
233 Enviar_char("00110000",r_data,r_disponible); — 0
234 Enviar_char("00110000",r_data,r_disponible); — 0
235 Enviar_char("00111110",r_data,r_disponible); — >
236
237 wait for periodo * TbPeriod;
238
239 — 21 ~ 21
240 N <= 23;
241 Enviar_char("00111100",r_data,r_disponible); — <
242 Enviar_char("00110001",r_data,r_disponible); — 1
243 Enviar_char("00110000",r_data,r_disponible); — 0
244 Enviar_char("00110000",r_data,r_disponible); — 0
245 Enviar_char("00110000",r_data,r_disponible); — 0
246 Enviar_char("00110000",r_data,r_disponible); — 0
247 Enviar_char("00110000",r_data,r_disponible); — 0
248 Enviar_char("00110000",r_data,r_disponible); — 0
249 Enviar_char("00110000",r_data,r_disponible); — 0
250 Enviar_char("00110000",r_data,r_disponible); — 0
251 Enviar_char("00110000",r_data,r_disponible); — 0
252 Enviar_char("00110000",r_data,r_disponible); — 0
253 Enviar_char("00110000",r_data,r_disponible); — 0
254 Enviar_char("00110000",r_data,r_disponible); — 0
255 Enviar_char("00110000",r_data,r_disponible); — 0
256 Enviar_char("00110000",r_data,r_disponible); — 0
257 Enviar_char("00110000",r_data,r_disponible); — 0
258 Enviar_char("00110000",r_data,r_disponible); — 0
259 Enviar_char("00110000",r_data,r_disponible); — 0
260 Enviar_char("00110000",r_data,r_disponible); — 0
261 Enviar_char("00110000",r_data,r_disponible); — 0
262 Enviar_char("00110000",r_data,r_disponible); — 0
263 Enviar_char("00111110",r_data,r_disponible); — >
264
265 wait for periodo * TbPeriod;
266
267 — 22
268 N <= 24;

```

54

## Capítulo 4. Ensayos y Resultados

```

269 Enviar_char("00111100",r_data,r_disponible); — <
270 Enviar_char("00110001",r_data,r_disponible); — 1
271 Enviar_char("00110000",r_data,r_disponible); — 0
272 Enviar_char("00110001",r_data,r_disponible); — 1
273 Enviar_char("00110000",r_data,r_disponible); — 0
274 Enviar_char("00110001",r_data,r_disponible); — 1
275 Enviar_char("00110000",r_data,r_disponible); — 0
276 Enviar_char("00110001",r_data,r_disponible); — 1
277 Enviar_char("00110000",r_data,r_disponible); — 0
278 Enviar_char("00110001",r_data,r_disponible); — 1
279 Enviar_char("00110000",r_data,r_disponible); — 0
280 Enviar_char("00110001",r_data,r_disponible); — 1
281 Enviar_char("00110000",r_data,r_disponible); — 0
282 Enviar_char("00110001",r_data,r_disponible); — 1
283 Enviar_char("00110000",r_data,r_disponible); — 0
284 Enviar_char("00110001",r_data,r_disponible); — 1
285 Enviar_char("00110000",r_data,r_disponible); — 0
286 Enviar_char("00110001",r_data,r_disponible); — 1
287 Enviar_char("00110000",r_data,r_disponible); — 0
288 Enviar_char("00110001",r_data,r_disponible); — 1
289 Enviar_char("00110000",r_data,r_disponible); — 0
290 Enviar_char("00110001",r_data,r_disponible); — 1
291 Enviar_char("00110000",r_data,r_disponible); — 0
292 Enviar_char("00111110",r_data,r_disponible); — >
293
294 wait for periodo * TbPeriod;
TbSimEnded <= '1';
end process;
295
296 end tb;
297
298

```

ALGORITMO 4.3: Testbench del módulo UART

## 4.4.2. Resultados obtenidos

En la figura 4.3 se presentan los datos del ensayo. Se puede ver que las tramas injectadas son presentadas idénticamente en la salida, con un pequeño delay temporal. Además de poder apreciar el tren de pulsos que envía la UART de un extremo al otro del módulo, primero para indicar la lectura de la FIFO de recepción y luego para indicar la escritura de la FIFO de transmisión.

Solo después de superado este ensayo se pudo avanzar con la inclusión de los demás módulos. No tenía ningún sentido implementar sistemas complejos sin la certeza de que la transmisión y recepción de datos de la computadora a la plataforma era fiable. Este proceso de desarrollo requirió varias iteraciones por la

```

196 Enviar_char("00110000",r_data,r_disponible); — 0
197 Enviar_char("00110000",r_data,r_disponible); — 0
198 Enviar_char("00110000",r_data,r_disponible); — 0
199 Enviar_char("00110001",r_data,r_disponible); — 1
200 Enviar_char("00110000",r_data,r_disponible); — 0
201 Enviar_char("00111110",r_data,r_disponible); — >
202
203 wait for periodo * TbPeriod;
204
205 — Moviendo el "1" en todas las posiciones —

```

## 4.4. Validación de la UART

55

```

206 — Borrado para la memoria para resumir —
207
208 — Moviendo el "1" en todas las posiciones —
209
210 — 21 ~ 20
211 N <= 23;
212 Enviar_char("00111100",r_data,r_disponible); — <
213 Enviar_char("00110000",r_data,r_disponible); — 0
214 Enviar_char("00110001",r_data,r_disponible); — 1
215 Enviar_char("00110000",r_data,r_disponible); — 0
216 Enviar_char("00110000",r_data,r_disponible); — 0
217 Enviar_char("00110000",r_data,r_disponible); — 0
218 Enviar_char("00110000",r_data,r_disponible); — 0
219 Enviar_char("00110000",r_data,r_disponible); — 0
220 Enviar_char("00110000",r_data,r_disponible); — 0
221 Enviar_char("00110000",r_data,r_disponible); — 0
222 Enviar_char("00110000",r_data,r_disponible); — 0
223 Enviar_char("00110000",r_data,r_disponible); — 0
224 Enviar_char("00110000",r_data,r_disponible); — 0
225 Enviar_char("00110000",r_data,r_disponible); — 0
226 Enviar_char("00110000",r_data,r_disponible); — 0
227 Enviar_char("00110000",r_data,r_disponible); — 0
228 Enviar_char("00110000",r_data,r_disponible); — 0
229 Enviar_char("00110000",r_data,r_disponible); — 0
230 Enviar_char("00110000",r_data,r_disponible); — 0
231 Enviar_char("00110000",r_data,r_disponible); — 0
232 Enviar_char("00110000",r_data,r_disponible); — 0
233 Enviar_char("00110000",r_data,r_disponible); — 0
234 Enviar_char("00111110",r_data,r_disponible); — >
235
236 wait for periodo * TbPeriod;
237
238 — 21 ~ 21
239 N <= 23;
240 Enviar_char("00111100",r_data,r_disponible); — <
241 Enviar_char("00110001",r_data,r_disponible); — 1
242 Enviar_char("00110000",r_data,r_disponible); — 0
243 Enviar_char("00110000",r_data,r_disponible); — 0
244 Enviar_char("00110000",r_data,r_disponible); — 0
245 Enviar_char("00110000",r_data,r_disponible); — 0
246 Enviar_char("00110000",r_data,r_disponible); — 0
247 Enviar_char("00110000",r_data,r_disponible); — 0
248 Enviar_char("00110000",r_data,r_disponible); — 0
249 Enviar_char("00110000",r_data,r_disponible); — 0
250 Enviar_char("00110000",r_data,r_disponible); — 0
251 Enviar_char("00110000",r_data,r_disponible); — 0
252 Enviar_char("00110000",r_data,r_disponible); — 0
253 Enviar_char("00110000",r_data,r_disponible); — 0
254 Enviar_char("00110000",r_data,r_disponible); — 0
255 Enviar_char("00110000",r_data,r_disponible); — 0
256 Enviar_char("00110000",r_data,r_disponible); — 0
257 Enviar_char("00110000",r_data,r_disponible); — 0
258 Enviar_char("00110000",r_data,r_disponible); — 0
259 Enviar_char("00110000",r_data,r_disponible); — 0
260 Enviar_char("00110000",r_data,r_disponible); — 0
261 Enviar_char("00110000",r_data,r_disponible); — 0
262 Enviar_char("00110000",r_data,r_disponible); — 0
263 Enviar_char("00111110",r_data,r_disponible); — >
264
265 wait for periodo * TbPeriod;
266
267 — 22
268 N <= 24;

```

## Capítulo 4. Ensayos y Resultados

```

269 Enviar_char("00111100",r_data,r_disponible); — <
270 Enviar_char("00110001",r_data,r_disponible); — 1
271 Enviar_char("00110000",r_data,r_disponible); — 0
272 Enviar_char("00110001",r_data,r_disponible); — 1
273 Enviar_char("00110000",r_data,r_disponible); — 0
274 Enviar_char("00110001",r_data,r_disponible); — 1
275 Enviar_char("00110000",r_data,r_disponible); — 0
276 Enviar_char("00110001",r_data,r_disponible); — 1
277 Enviar_char("00110000",r_data,r_disponible); — 0
278 Enviar_char("00110001",r_data,r_disponible); — 1
279 Enviar_char("00110000",r_data,r_disponible); — 0
280 Enviar_char("00110001",r_data,r_disponible); — 1
281 Enviar_char("00110000",r_data,r_disponible); — 0
282 Enviar_char("00110001",r_data,r_disponible); — 1
283 Enviar_char("00110000",r_data,r_disponible); — 0
284 Enviar_char("00110001",r_data,r_disponible); — 1
285 Enviar_char("00110000",r_data,r_disponible); — 0
286 Enviar_char("00110001",r_data,r_disponible); — 1
287 Enviar_char("00110000",r_data,r_disponible); — 0
288 Enviar_char("00110001",r_data,r_disponible); — 1
289 Enviar_char("00110000",r_data,r_disponible); — 0
290 Enviar_char("00110001",r_data,r_disponible); — 1
291 Enviar_char("00110000",r_data,r_disponible); — 0
292 Enviar_char("00111110",r_data,r_disponible); — >
293
294 wait for periodo * TbPeriod;
TbSimEnded <= '1';
end process;
295
296 end tb;
297
298

```

ALGORITMO 4.3: Testbench del módulo UART

## 4.4.2. Resultados obtenidos

En la figura 4.3 se presentan los datos del ensayo. Se puede ver que las tramas injectadas son presentadas idénticamente en la salida, con un pequeño delay temporal. Además de poder apreciar el tren de pulsos que envía la UART de un extremo al otro del módulo, primero para indicar la lectura de la FIFO de recepción y luego para indicar la escritura de la FIFO de transmisión.

Solo después de superado este ensayo se pudo avanzar con la inclusión de los demás módulos. No tenía ningún sentido implementar sistemas complejos sin la certeza de que la transmisión y recepción de datos de la computadora a la plataforma era fiable. Este proceso de desarrollo requirió varias iteraciones por la

anotación que implica depurar cada uno de los elementos involucrados.

#### 4.5. Validación del detector

El bloque detector tiene como función descartar todas las tramas que no cumplen con el requisito definido de poseer tag inicial y final, además de que el mensaje debe tener una cantidad de elementos indicado por la UART.

Para probar el sistema se deberán enviar:

- Tramas con tags incorrectos

##### 4.5. Validación del detector

55

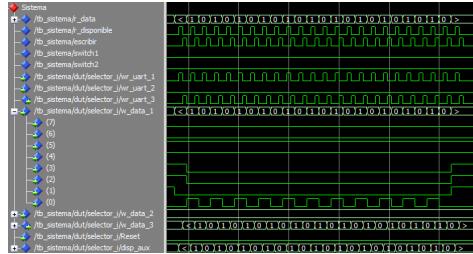


FIGURA 4.3: Simulación de la UART.

- Tramas con ausencia de alguno o ambos tags
- Tramas con menos datos de los indicados
- Tramas con más datos de los indicados
- Tramas correctas de elementos nulos, con un solo elemento válido en todas las posiciones posibles.

##### 4.5.1. Testbench del módulo detector

Se generó el Algoritmo 4.4 que inyecta las señales necesarias para probar la detección de las tramas correctas y el descarte de las incorrectas. Además deberá enviar las señales de activación/desactivación de las próximas etapas, así como recibir las señales inhibidoras necesarias. Por una cuestión de espacio se recortó parte del algoritmo presentado.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity tb_detector is
5  end tb_detector;
6
7
8  architecture tb of tb_detector is
9
10 component detector
11  port(
12    clk_i: in std_logic;
13    rst_i: in std_logic;
14    r_data: in std_logic_vector(8-1 downto 0);
15    r_disponible : in std_logic;
16    led_rgb_1 : out std_logic_vector(3-1 downto 0);
17    led_rgb_2 : out std_logic_vector(3-1 downto 0);
18    paquete: out std_logic_vector(21-1 downto 0);
19    procesar : in std_logic;
20    procesado : out std_logic;
21    N : in integer;
22    N_1 : out integer;
23    N_2 : out integer;

```

56

##### Capítulo 4. Ensayos y Resultados

```

24  wr_uart : out std_logic;
25  w_data: out std_logic_vector(8-1 downto 0)
26  );
27 end component;
28
29 signal clk_i      : std_logic;
30 signal rst_i      : std_logic;
31 signal r_data     : std_logic_vector (8-1 downto 0);
32 signal r_disponible : std_logic;
33 signal led_rgb_1  : std_logic_vector (3-1 downto 0);
34 signal led_rgb_2  : std_logic_vector (3-1 downto 0);
35 signal paquete   : std_logic_vector (21-1 downto 0);
36 signal procesar   : std_logic;
37 signal procesado  : std_logic;
38 signal N,N_1,N_2 : integer;
39 signal wr_uart   : std_logic;
40 signal w_data     : std_logic_vector (8-1 downto 0);
41
42 constant TbPeriod : time := 8 ns; — EDIT Put right period here
43 signal TbClock : std_logic := '0';
44 signal TbSimEnded : std_logic := '0';
45
46 — Low-level byte-write
47 procedure Enviar_char (
48   data : in std_logic_vector(8-1 downto 0);
49   signal r_data : out std_logic_vector(8-1 downto 0);
50   signal r_disponible : out std_logic) is
51 begin
52
53   r_disponible <= '0';
54   wait for TbPeriod;
55   r_data <= data;
56   wait for TbPeriod;
57   r_disponible <= '1';
58   wait for TbPeriod;
59   r_disponible <= '0';
60
61   end Enviar_char;
62
63 begin
64
65   dut : detector
66   port map (clk_i      => clk_i,
67             rst_i      => rst_i,
68             r_data     => r_data,
69             r_disponible => r_disponible,
70             led_rgb_1  => led_rgb_1,

```

##### 4.5. Validación del detector

anotación que implica depurar cada uno de los elementos involucrados.

#### 4.5. Validación del detector

El bloque detector tiene como función descartar todas las tramas que no cumplen con el requisito definido de poseer tag inicial y final, además de que el mensaje debe tener una cantidad de elementos indicado por la UART.

Para probar el sistema se deberán enviar:

- Tramas con tags incorrectos

##### 4.5. Validación del detector

57

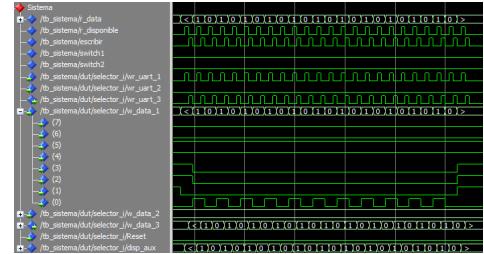


FIGURA 4.3: Simulación de la UART.

- Tramas con ausencia de alguno o ambos tags
- Tramas con menos datos de los indicados
- Tramas con más datos de los indicados
- Tramas correctas de elementos nulos, con un solo elemento válido en todas las posiciones posibles.

##### 4.5.1. Testbench del módulo detector

Se generó el Algoritmo 4.4 que inyecta las señales necesarias para probar la detección de las tramas correctas y el descarte de las incorrectas. Además deberá enviar las señales de activación/desactivación de las próximas etapas, así como recibir las señales inhibidoras necesarias. Por una cuestión de espacio se recortó parte del algoritmo presentado.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity tb_detector is
5  end tb_detector;
6
7
8  architecture tb of tb_detector is
9
10 component detector
11  port(
12    clk_i: in std_logic;
13    rst_i: in std_logic;
14    r_data: in std_logic_vector(8-1 downto 0);
15    r_disponible : in std_logic;
16    led_rgb_1 : out std_logic_vector(3-1 downto 0);
17    led_rgb_2 : out std_logic_vector(3-1 downto 0);
18    paquete: out std_logic_vector(21-1 downto 0);
19    procesar : in std_logic;
20    procesado : out std_logic;
21    N : in integer;
22    N_1 : out integer;
23    N_2 : out integer;

```

58

##### Capítulo 4. Ensayos y Resultados

```

24  wr_uart : out std_logic;
25  w_data: out std_logic_vector(8-1 downto 0)
26  );
27 end component;
28
29 signal clk_i      : std_logic;
30 signal rst_i      : std_logic;
31 signal r_data     : std_logic_vector (8-1 downto 0);
32 signal r_disponible : std_logic;
33 signal led_rgb_1  : std_logic_vector (3-1 downto 0);
34 signal led_rgb_2  : std_logic_vector (3-1 downto 0);
35 signal paquete   : std_logic_vector (21-1 downto 0);
36 signal procesar   : std_logic;
37 signal procesado  : std_logic;
38 signal N,N_1,N_2 : integer;
39 signal wr_uart   : std_logic;
40 signal w_data     : std_logic_vector (8-1 downto 0);
41
42 constant TbPeriod : time := 8 ns; — EDIT Put right period here
43 signal TbClock : std_logic := '0';
44 signal TbSimEnded : std_logic := '0';
45
46 — Low-level byte-write
47 procedure Enviar_char (
48   data : in std_logic_vector(8-1 downto 0);
49   signal r_data : out std_logic_vector(8-1 downto 0);
50   signal r_disponible : out std_logic) is
51 begin
52
53   r_disponible <= '0';
54   wait for TbPeriod;
55   r_data <= data;
56   wait for TbPeriod;
57   r_disponible <= '1';
58   wait for TbPeriod;
59   r_disponible <= '0';
60
61   end Enviar_char;
62
63 begin
64
65   dut : detector
66   port map (clk_i      => clk_i,
67             rst_i      => rst_i,
68             r_data     => r_data,
69             r_disponible => r_disponible,
70             led_rgb_1  => led_rgb_1,

```

```

71     led_rgb_2 => led_rgb_2,
72     paquete => paquete,
73   procesar => procesar,
74   procesado => procesado,
75 N => N,
76   N_1 => N_1,
77 N_2 => N_2,
78 wr_uart => wr_uart,
79   w_data => w_data);
80
81 — Clock generation
82 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
83   '0';
84
85 — EDIT: Check that clk_i is really your main clock signal
86 clk_i <= TbClock;

```

## 4.5. Validación del detector

57

```

86 stimuli : process
87 begin
88   — EDIT Adapt initialization as needed
89   —r_data <= (others => '0');
90
91   — Reset generation
92   — EDIT: Check that rst_i is really your reset signal
93   rst_i <= '1';
94   wait for 100 ns;
95   rst_i <= '0';
96
97
98   wait for 1000 * TbPeriod;
100
101  — Stop the clock and hence terminate the simulation
102 TbSimEnded <= '1';
103  wait;
104 end process;
105
106 datos : process
107 begin
108   r_data <= "00000000";
109   r_disponible <= '0';
110
111   wait for 100 ns;
112
113 — 21
114 N <= 23;
115 Enviar_char("00111100",r_data,r_disponible); — <
116 Enviar_char("00110001",r_data,r_disponible); — 1
117 Enviar_char("00110001",r_data,r_disponible); — 1
118 Enviar_char("00110001",r_data,r_disponible); — 1
119 Enviar_char("00110001",r_data,r_disponible); — 1
120 Enviar_char("00110000",r_data,r_disponible); — 0
121 Enviar_char("00110001",r_data,r_disponible); — 1
122 Enviar_char("00110000",r_data,r_disponible); — 0
123 Enviar_char("00110001",r_data,r_disponible); — 1
124 Enviar_char("00110000",r_data,r_disponible); — 0
125 Enviar_char("00110001",r_data,r_disponible); — 1
126 Enviar_char("00110000",r_data,r_disponible); — 0
127 Enviar_char("00110001",r_data,r_disponible); — 1
128 Enviar_char("00110000",r_data,r_disponible); — 0
129 Enviar_char("00110001",r_data,r_disponible); — 1
130 Enviar_char("00110000",r_data,r_disponible); — 0
131 Enviar_char("00110001",r_data,r_disponible); — 1
132 Enviar_char("00110000",r_data,r_disponible); — 0
133 Enviar_char("00110001",r_data,r_disponible); — 1
134 Enviar_char("00110000",r_data,r_disponible); — 0
135 Enviar_char("00110000",r_data,r_disponible); — 0
136 Enviar_char("00110001",r_data,r_disponible); — 1
137 Enviar_char("00110001",r_data,r_disponible); — 1
138 Enviar_char("00111110",r_data,r_disponible); — >
139
140 wait for 100 * TbPeriod;
141
142 — 21
143 N <= 23;
144 Enviar_char("00111100",r_data,r_disponible); — <
145 Enviar_char("00110001",r_data,r_disponible); — 1
146 Enviar_char("00110001",r_data,r_disponible); — 1
147 Enviar_char("00110001",r_data,r_disponible); — 1
148 Enviar_char("00110000",r_data,r_disponible); — 0

```

58

## Capítulo 4. Ensayos y Resultados

```

149 Enviar_char("00110001",r_data,r_disponible); — 1
150 Enviar_char("00110000",r_data,r_disponible); — 0
151 Enviar_char("00110001",r_data,r_disponible); — 1
152 Enviar_char("00110000",r_data,r_disponible); — 0
153 Enviar_char("00110001",r_data,r_disponible); — 1
154 Enviar_char("00110000",r_data,r_disponible); — 0
155 Enviar_char("00110001",r_data,r_disponible); — 1
156 Enviar_char("00110000",r_data,r_disponible); — 0
157 Enviar_char("00110001",r_data,r_disponible); — 1
158 Enviar_char("00110000",r_data,r_disponible); — 0
159 Enviar_char("00110001",r_data,r_disponible); — 1
160 Enviar_char("00110000",r_data,r_disponible); — 0
161 Enviar_char("00110001",r_data,r_disponible); — 1
162 Enviar_char("00110000",r_data,r_disponible); — 0
163 Enviar_char("00110000",r_data,r_disponible); — 0
164 Enviar_char("00110001",r_data,r_disponible); — 1
165 Enviar_char("00110001",r_data,r_disponible); — 1
166 Enviar_char("00111110",r_data,r_disponible); — >
167
168 wait for 100 * TbPeriod;
169
170 — 22
171 N <= 24;
172 Enviar_char("00111100",r_data,r_disponible); — <
173 Enviar_char("00110001",r_data,r_disponible); — 1
174 Enviar_char("00110000",r_data,r_disponible); — 0
175 Enviar_char("00110001",r_data,r_disponible); — 1
176 Enviar_char("00110000",r_data,r_disponible); — 0
177 Enviar_char("00110001",r_data,r_disponible); — 1
178 Enviar_char("00110000",r_data,r_disponible); — 0
179 Enviar_char("00110001",r_data,r_disponible); — 1
180 Enviar_char("00110000",r_data,r_disponible); — 0
181 Enviar_char("00110001",r_data,r_disponible); — 1
182 Enviar_char("00110000",r_data,r_disponible); — 0
183 Enviar_char("00110001",r_data,r_disponible); — 1
184 Enviar_char("00110000",r_data,r_disponible); — 0
185 Enviar_char("00110001",r_data,r_disponible); — 1
186 Enviar_char("00110000",r_data,r_disponible); — 0
187 Enviar_char("00110001",r_data,r_disponible); — 1
188 Enviar_char("00110000",r_data,r_disponible); — 0
189 Enviar_char("00110001",r_data,r_disponible); — 1
190 Enviar_char("00110000",r_data,r_disponible); — 0
191 Enviar_char("00110001",r_data,r_disponible); — 1
192 Enviar_char("00110000",r_data,r_disponible); — 0

```

60

## Capítulo 4. Ensayos y Resultados

```

149 Enviar_char("00110001",r_data,r_disponible); — 1
150 Enviar_char("00110000",r_data,r_disponible); — 0
151 Enviar_char("00110001",r_data,r_disponible); — 1
152 Enviar_char("00110000",r_data,r_disponible); — 0
153 Enviar_char("00110001",r_data,r_disponible); — 1
154 Enviar_char("00110000",r_data,r_disponible); — 0
155 Enviar_char("00110001",r_data,r_disponible); — 1
156 Enviar_char("00110000",r_data,r_disponible); — 0
157 Enviar_char("00110001",r_data,r_disponible); — 1
158 Enviar_char("00110000",r_data,r_disponible); — 0
159 Enviar_char("00110001",r_data,r_disponible); — 1
160 Enviar_char("00110000",r_data,r_disponible); — 0
161 Enviar_char("00110001",r_data,r_disponible); — 1
162 Enviar_char("00110000",r_data,r_disponible); — 0
163 Enviar_char("00110000",r_data,r_disponible); — 0
164 Enviar_char("00110001",r_data,r_disponible); — 1
165 Enviar_char("00110001",r_data,r_disponible); — 1
166 Enviar_char("00111110",r_data,r_disponible); — >
167
168 wait for 100 * TbPeriod;
169
170 — 22
171 N <= 24;
172 Enviar_char("00111100",r_data,r_disponible); — <
173 Enviar_char("00110001",r_data,r_disponible); — 1
174 Enviar_char("00110000",r_data,r_disponible); — 0
175 Enviar_char("00110001",r_data,r_disponible); — 1
176 Enviar_char("00110000",r_data,r_disponible); — 0
177 Enviar_char("00110001",r_data,r_disponible); — 1
178 Enviar_char("00110000",r_data,r_disponible); — 0
179 Enviar_char("00110001",r_data,r_disponible); — 1
180 Enviar_char("00110000",r_data,r_disponible); — 0
181 Enviar_char("00110001",r_data,r_disponible); — 1
182 Enviar_char("00110000",r_data,r_disponible); — 0
183 Enviar_char("00110001",r_data,r_disponible); — 1
184 Enviar_char("00110000",r_data,r_disponible); — 0
185 Enviar_char("00110001",r_data,r_disponible); — 1
186 Enviar_char("00110000",r_data,r_disponible); — 0
187 Enviar_char("00110001",r_data,r_disponible); — 1
188 Enviar_char("00110000",r_data,r_disponible); — 0
189 Enviar_char("00110001",r_data,r_disponible); — 1
190 Enviar_char("00110000",r_data,r_disponible); — 0
191 Enviar_char("00110001",r_data,r_disponible); — 1
192 Enviar_char("00110000",r_data,r_disponible); — 0

```



```

24 led_rgb_1 : out std_logic_vector(3-1 downto 0);
25 led_rgb_2 : out std_logic_vector(3-1 downto 0);
26 w_data: out std_logic_vector(8-1 downto 0)
27 );
28 end component;
29
30
31 signal Clock : std_logic;
32 signal Reset : std_logic;
33 signal r_data : std_logic_vector (8-1 downto 0);
34 signal r_disponible : std_logic;
35 signal leer : std_logic;
36 signal escribir : std_logic;
37 signal switch1 : std_logic;
38 signal switch2 : std_logic;
39 signal reset_uart : std_logic;
40 signal N : integer;
41 signal leds : std_logic_vector(4-1 downto 0);
42 signal led_rgb_1 : std_logic_vector (3-1 downto 0);
43 signal led_rgb_2 : std_logic_vector (3-1 downto 0);
44

```

## 4.6. Validación del enclavamiento

61

```

45 signal w_data : std_logic_vector (8-1 downto 0);
46
47 constant TbPeriod : time := 8 ns; — EDIT Put right period here
48 signal TbClock : std_logic := '0';
49 signal TbSimEnded : std_logic := '0';
50
51 constant periodo : integer := 100;
52
53 — Low-level byte-write
54 procedure Enviar_char (
55   data : in std_logic_vector(8-1 downto 0);
56   signal r_data : out std_logic_vector(8-1 downto 0);
57   signal r_disponible : out std_logic) is
58 begin
59
60   r_disponible <= '0';
61   wait for TbPeriod;
62   r_data <= data;
63   wait for TbPeriod;
64   r_disponible <= '1';
65   wait for TbPeriod;
66   r_disponible <= '0';
67
68   end Enviar_char;
69
70 begin
71
72   dut : sistema
73   port map (Clock      => Clock,
74             Reset       => Reset,
75             r_data      => r_data,
76             r_disponible => r_disponible,
77             leer        => leer,
78             escribir    => escribir,
79             switch1     => switch1,
80             switch2     => switch2,
81             reset_uart  => reset_uart,
82             N           => N,
83             leds        => leds,
84             led_rgb_1   => led_rgb_1,
85             led_rgb_2   => led_rgb_2,
86             w_data      => w_data);
87
88 — Clock generation
89 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
90 '0';
91
92 — EDIT: Check that clk_i is really your main clock signal
93 Clock <= TbClock;
94
95 stimuli : process
96 begin
97   — EDIT Adapt initialization as needed
98   —r_data <= (others => '0');
99
100  — Reset generation
101  — EDIT: Check that rst_i is really your reset signal
102  switch1 <= '1';
103  switch2 <= '1';
104  Reset <= '1';
105  wait for 100 ns;
106  Reset <= '0';
107  switch1 <= '1';

```

62

## Capítulo 4. Ensayos y Resultados

```

107 switch2 <= '1';
108 wait for 250000 * TbPeriod;
109
110 — Stop the clock and hence terminate the simulation
111 TbSimEnded <= '1';
112 wait;
113 end process;
114
115 datos : process
116 begin
117
118 r_data <= "00000000";
119 r_disponible <= '0';
120
121 wait for 100 ns;
122
123 — 21 – 21 – todos 1
124 N <= 23;
125 Enviar_char("00111100",r_data,r_disponible); — <
126 Enviar_char("00110001",r_data,r_disponible); — 1
127 Enviar_char("00110001",r_data,r_disponible); — 1
128 Enviar_char("00110001",r_data,r_disponible); — 1
129 Enviar_char("00110001",r_data,r_disponible); — 1
130 Enviar_char("00110001",r_data,r_disponible); — 1
131 Enviar_char("00110001",r_data,r_disponible); — 1
132 Enviar_char("00110001",r_data,r_disponible); — 1
133 Enviar_char("00110001",r_data,r_disponible); — 1
134 Enviar_char("00110001",r_data,r_disponible); — 1
135 Enviar_char("00110001",r_data,r_disponible); — 1
136 Enviar_char("00110001",r_data,r_disponible); — 1
137 Enviar_char("00110001",r_data,r_disponible); — 1
138 Enviar_char("00110001",r_data,r_disponible); — 1
139 Enviar_char("00110001",r_data,r_disponible); — 1
140 Enviar_char("00110001",r_data,r_disponible); — 1
141 Enviar_char("00110001",r_data,r_disponible); — 1
142 Enviar_char("00110001",r_data,r_disponible); — 1
143 Enviar_char("00110001",r_data,r_disponible); — 1
144 Enviar_char("00110001",r_data,r_disponible); — 1
145 Enviar_char("00110001",r_data,r_disponible); — 1

```

## 4.6. Validación del enclavamiento

63

```

45 signal w_data : std_logic_vector (8-1 downto 0);
46
47 constant TbPeriod : time := 8 ns; — EDIT Put right period here
48 signal TbClock : std_logic := '0';
49 signal TbSimEnded : std_logic := '0';
50
51 constant periodo : integer := 100;
52
53 — Low-level byte-write
54 procedure Enviar_char (
55   data : in std_logic_vector(8-1 downto 0);
56   signal r_data : out std_logic_vector(8-1 downto 0);
57   signal r_disponible : out std_logic) is
58 begin
59
60   r_disponible <= '0';
61   wait for TbPeriod;
62   r_data <= data;
63   wait for TbPeriod;
64   r_disponible <= '1';
65   wait for TbPeriod;
66   r_disponible <= '0';
67
68   end Enviar_char;
69
70 begin
71
72   dut : sistema
73   port map (Clock      => Clock,
74             Reset       => Reset,
75             r_data      => r_data,
76             r_disponible => r_disponible,
77             leer        => leer,
78             escribir    => escribir,
79             switch1     => switch1,
80             switch2     => switch2,
81             reset_uart  => reset_uart,
82             N           => N,
83             leds        => leds,
84             led_rgb_1   => led_rgb_1,
85             led_rgb_2   => led_rgb_2,
86             w_data      => w_data);
87
88 — Clock generation
89 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
90 '0';
91
92 — EDIT: Check that clk_i is really your main clock signal
93 Clock <= TbClock;
94
95 stimuli : process
96 begin
97   — EDIT Adapt initialization as needed
98   —r_data <= (others => '0');
99
100  — Reset generation
101  — EDIT: Check that rst_i is really your reset signal
102  switch1 <= '1';
103  switch2 <= '1';
104  Reset <= '1';
105  wait for 100 ns;
106  Reset <= '0';
107  switch1 <= '1';

```

64

## Capítulo 4. Ensayos y Resultados

```

107 switch2 <= '1';
108 wait for 250000 * TbPeriod;
109
110 — Stop the clock and hence terminate the simulation
111 TbSimEnded <= '1';
112 wait;
113 end process;
114
115 datos : process
116 begin
117
118 r_data <= "00000000";
119 r_disponible <= '0';
120
121 wait for 100 ns;
122
123 — 21 – 21 – todos 1
124 N <= 23;
125 Enviar_char("00111100",r_data,r_disponible); — <
126 Enviar_char("00110001",r_data,r_disponible); — 1
127 Enviar_char("00110001",r_data,r_disponible); — 1
128 Enviar_char("00110001",r_data,r_disponible); — 1
129 Enviar_char("00110001",r_data,r_disponible); — 1
130 Enviar_char("00110001",r_data,r_disponible); — 1
131 Enviar_char("00110001",r_data,r_disponible); — 1
132 Enviar_char("00110001",r_data,r_disponible); — 1
133 Enviar_char("00110001",r_data,r_disponible); — 1
134 Enviar_char("00110001",r_data,r_disponible); — 1
135 Enviar_char("00110001",r_data,r_disponible); — 1
136 Enviar_char("00110001",r_data,r_disponible); — 1
137 Enviar_char("00110001",r_data,r_disponible); — 1
138 Enviar_char("00110001",r_data,r_disponible); — 1
139 Enviar_char("00110001",r_data,r_disponible); — 1
140 Enviar_char("00110001",r_data,r_disponible); — 1
141 Enviar_char("00110001",r_data,r_disponible); — 1
142 Enviar_char("00110001",r_data,r_disponible); — 1
143 Enviar_char("00110001",r_data,r_disponible); — 1
144 Enviar_char("00110001",r_data,r_disponible); — 1
145 Enviar_char("00110001",r_data,r_disponible); — 1

```







```

25
26 signal clk_i, rst_i, switch, wr_uart_3, r_disponible : std_logic;
27 signal leds : std_logic_vector(2-1 downto 0);
28 signal N_1,N_2 : integer;
29 signal w_data_1,w_data_2,w_data_3 : std_logic_vector(8-1 downto 0);
30
31 constant TbPeriod : time := 8 ns; — EDIT Put right period here
32 signal TbClock : std_logic := '0';
33 signal TbSimEnded : std_logic := '0';
34
35 — Low-level byte-write
36 procedure Enviar_char (
37   data : in std_logic_vector(8-1 downto 0);
38   signal r_data : out std_logic_vector(8-1 downto 0);
39   signal r_disponible : out std_logic) is
40 begin
41   r_disponible <= '0';
42   wait for TbPeriod;
43   r_data <= data;
44   wait for 407 * TbPeriod;
45   r_disponible <= '1';
46   wait for TbPeriod;
47   r_disponible <= '0';
48
49 end Enviar_char;
50
51 begin
52   dut : connector_test
53   port map (clk_i => clk_i,
54             rst_i => rst_i,
55             switch => switch,
56
57

```

## 4.8. Validación del selector

69

```

58   leds => leds,
59   wr_uart_3 => wr_uart_3,
60   N_1 => N_1,
61   N_2 => N_2,
62   r_disponible => r_disponible,
63   w_data_1 => w_data_1,
64   w_data_2 => w_data_2,
65   w_data_3 => w_data_3);
66
67 — Clock generation
68 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
69   '0';
70
71 — EDIT: Check that clk_i is really your main clock signal
72 clk_i <= TbClock;
73
74 stimuli : process
75 begin
76   — EDIT Adapt initialization as needed
77   —r_data <= (others => '0');
78
79   — Reset generation
80   — EDIT: Check that rst_i is really your reset signal
81   rst_i <= '1';
82   wait for 100 ns;
83   rst_i <= '0';
84
85   wait for 1000000 * TbPeriod;
86
87   — Stop the clock and hence terminate the simulation
88 TbSimEnded <= '1';
89   wait;
90 end process;
91
92 datos : process
93 begin
94
95   w_data_2 <= "00000000";
96   N_2 <= 0;
97   switch <= '0';
98
99   wait for 200 ns;
100
101 — 21
102 N_1 <= 23;
103 r_disponible <= '1';
104 —wait for 5 * TbPeriod;
105 Enviar_char("00111100","w_data_1,r_disponible"); — <
106 Enviar_char("00110001","w_data_1,r_disponible"); — 1
107 Enviar_char("00110001","w_data_1,r_disponible"); — 1
108 Enviar_char("00110001","w_data_1,r_disponible"); — 1
109 Enviar_char("00110000","w_data_1,r_disponible"); — 0
110 Enviar_char("00110000","w_data_1,r_disponible"); — 1
111 Enviar_char("00110000","w_data_1,r_disponible"); — 0
112 Enviar_char("00110000","w_data_1,r_disponible"); — 1
113 Enviar_char("00110000","w_data_1,r_disponible"); — 0
114 Enviar_char("00110000","w_data_1,r_disponible"); — 1
115 Enviar_char("00110000","w_data_1,r_disponible"); — 0
116 Enviar_char("00110000","w_data_1,r_disponible"); — 1
117 Enviar_char("00110000","w_data_1,r_disponible"); — 0
118 Enviar_char("00110001","w_data_1,r_disponible"); — 1
119 Enviar_char("00110000","w_data_1,r_disponible"); — 0

```

70

## Capítulo 4. Ensayos y Resultados

```

120 Enviar_char("00110001","w_data_1,r_disponible"); — 1
121 Enviar_char("00110000","w_data_1,r_disponible"); — 0
122 Enviar_char("00110001","w_data_1,r_disponible"); — 1
123 Enviar_char("00110000","w_data_1,r_disponible"); — 0
124 Enviar_char("00110000","w_data_1,r_disponible"); — 0
125 Enviar_char("00110001","w_data_1,r_disponible"); — 1
126 Enviar_char("00110001","w_data_1,r_disponible"); — 1
127 Enviar_char("00111100","w_data_1,r_disponible"); — >
128 —wait for 5 * TbPeriod;
129 r_disponible <= '0';
130
131 switch <= '1';
132 r_disponible <= '1';
133 Enviar_char("00111100","w_data_1,r_disponible"); — <
134 Enviar_char("00110001","w_data_1,r_disponible"); — 1
135 Enviar_char("00110001","w_data_1,r_disponible"); — 1
136 Enviar_char("00110001","w_data_1,r_disponible"); — 1
137 Enviar_char("00110000","w_data_1,r_disponible"); — 0
138 Enviar_char("00110001","w_data_1,r_disponible"); — 1
139 Enviar_char("00110000","w_data_1,r_disponible"); — 0
140 Enviar_char("00110000","w_data_1,r_disponible"); — 1
141 Enviar_char("00110000","w_data_1,r_disponible"); — 0
142 Enviar_char("00110001","w_data_1,r_disponible"); — 1
143 Enviar_char("00110000","w_data_1,r_disponible"); — 0
144 Enviar_char("00110001","w_data_1,r_disponible"); — 1
145 Enviar_char("00110000","w_data_1,r_disponible"); — 0
146 Enviar_char("00110001","w_data_1,r_disponible"); — 1

```

```

25
26 signal clk_i, rst_i, switch, wr_uart_3, r_disponible : std_logic;
27 signal leds : std_logic_vector(2-1 downto 0);
28 signal N_1,N_2 : integer;
29 signal w_data_1,w_data_2,w_data_3 : std_logic_vector(8-1 downto 0);
30
31 constant TbPeriod : time := 8 ns; — EDIT Put right period here
32 signal TbClock : std_logic := '0';
33 signal TbSimEnded : std_logic := '0';
34
35 — Low-level byte-write
36 procedure Enviar_char (
37   data : in std_logic_vector(8-1 downto 0);
38   signal r_data : out std_logic_vector(8-1 downto 0);
39   signal r_disponible : out std_logic) is
40 begin
41   r_disponible <= '0';
42   wait for TbPeriod;
43   r_data <= data;
44   wait for 407 * TbPeriod;
45   r_disponible <= '1';
46   wait for TbPeriod;
47   r_disponible <= '0';
48
49 end Enviar_char;
50
51 begin
52   dut : connector_test
53   port map (clk_i => clk_i,
54             rst_i => rst_i,
55             switch => switch,
56
57

```

## 4.8. Validación del selector

71

```

58   leds => leds,
59   wr_uart_3 => wr_uart_3,
60   N_1 => N_1,
61   N_2 => N_2,
62   r_disponible => r_disponible,
63   w_data_1 => w_data_1,
64   w_data_2 => w_data_2,
65   w_data_3 => w_data_3);
66
67 — Clock generation
68 TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else
69   '0';
70
71 — EDIT: Check that clk_i is really your main clock signal
72 clk_i <= TbClock;
73
74 stimuli : process
75 begin
76   — EDIT Adapt initialization as needed
77   —r_data <= (others => '0');
78
79   — Reset generation
80   — EDIT: Check that rst_i is really your reset signal
81   rst_i <= '1';
82   wait for 100 ns;
83   rst_i <= '0';
84
85   wait for 1000000 * TbPeriod;
86
87   — Stop the clock and hence terminate the simulation
88 TbSimEnded <= '1';
89   wait;
90 end process;
91
92 datos : process
93 begin
94
95   w_data_2 <= "00000000";
96   N_2 <= 0;
97   switch <= '0';
98
99   wait for 200 ns;
100
101 — 21
102 N_1 <= 23;
103 r_disponible <= '1';
104 —wait for 5 * TbPeriod;
105 Enviar_char("00111100","w_data_1,r_disponible"); — <
106 Enviar_char("00110001","w_data_1,r_disponible"); — 1
107 Enviar_char("00110001","w_data_1,r_disponible"); — 1
108 Enviar_char("00110001","w_data_1,r_disponible"); — 1
109 Enviar_char("00110000","w_data_1,r_disponible"); — 0
110 Enviar_char("00110000","w_data_1,r_disponible"); — 1
111 Enviar_char("00110000","w_data_1,r_disponible"); — 0
112 Enviar_char("00110000","w_data_1,r_disponible"); — 1
113 Enviar_char("00110000","w_data_1,r_disponible"); — 0
114 Enviar_char("00110000","w_data_1,r_disponible"); — 1
115 Enviar_char("00110000","w_data_1,r_disponible"); — 0
116 Enviar_char("00110001","w_data_1,r_disponible"); — 1
117 Enviar_char("00110000","w_data_1,r_disponible"); — 0
118 Enviar_char("00110001","w_data_1,r_disponible"); — 1
119 Enviar_char("00110000","w_data_1,r_disponible"); — 0

```

72

## Capítulo 4. Ensayos y Resultados

```

120 Enviar_char("00110001","w_data_1,r_disponible"); — 1
121 Enviar_char("00110000","w_data_1,r_disponible"); — 0
122 Enviar_char("00110001","w_data_1,r_disponible"); — 1
123 Enviar_char("00110000","w_data_1,r_disponible"); — 0
124 Enviar_char("00110000","w_data_1,r_disponible"); — 0
125 Enviar_char("00110001","w_data_1,r_disponible"); — 1
126 Enviar_char("00110001","w_data_1,r_disponible"); — 1
127 Enviar_char("00111100","w_data_1,r_disponible"); — >
128 —wait for 5 * TbPeriod;
129 r_disponible <= '0';
130
131 switch <= '1';
132 r_disponible <= '1';
133 Enviar_char("00111100","w_data_1,r_disponible"); — <
134 Enviar_char("00110001","w_data_1,r_disponible"); — 1
135 Enviar_char("00110001","w_data_1,r_disponible"); — 1
136 Enviar_char("00110001","w_data_1,r_disponible"); — 1
137 Enviar_char("00110000","w_data_1,r_disponible"); — 0
138 Enviar_char("00110001","w_data_1,r_disponible"); — 1
139 Enviar_char("00110000","w_data_1,r_disponible"); — 0
140 Enviar_char("00110000","w_data_1,r_disponible"); — 1
141 Enviar_char("00110000","w_data_1,r_disponible"); — 0
142 Enviar_char("00110001","w_data_1,r_disponible"); — 1
143 Enviar_char("00110000","w_data_1,r_disponible"); — 0
144 Enviar_char("00110001","w_data_1,r_disponible"); — 1
145 Enviar_char("00110000","w_data_1,r_disponible"); — 0
146 Enviar_char("00110001","w_data_1,r_disponible"); — 1

```

```

147 Enviar_char("00110000",w_data_1,r_disponible); — 0
148 Enviar_char("00110001",w_data_1,r_disponible); — 1
149 Enviar_char("00110000",w_data_1,r_disponible); — 0
150 Enviar_char("00110001",w_data_1,r_disponible); — 1
151 Enviar_char("00110000",w_data_1,r_disponible); — 0
152 Enviar_char("00110000",w_data_1,r_disponible); — 0
153 Enviar_char("00110001",w_data_1,r_disponible); — 1
154 Enviar_char("00110001",w_data_1,r_disponible); — 1
155 Enviar_char("00111110",w_data_1,r_disponible); — >
156 —wait for 5 * TbPeriod;
157 r_disponible <= '0';
158
159 wait for 1000 * TbPeriod;
160
161 —wait for 100 * TbPeriod;
162 —TbSimEnded <= '1';
163 end process;
164
165 end tb;

```

ALGORITMO 4.7: Testbench del módulo selector

#### 4.8.2. Resultados obtenidos

En la figura 4.9 se visualiza el comportamiento del sistema con el switch en estado alto. Se puede comprobar que ambas señales llegan al módulo: w\_data\_1 y wr\_uart\_1 son la señal con información y el tren de pulsos que corresponden a no procesar la señal; mientras que w\_data\_2 y wr\_uart\_2 son las mismas pero luego de haber sido procesadas por el sistema de enclavamiento.

```

147 Enviar_char("00110000",w_data_1,r_disponible); — 0
148 Enviar_char("00110001",w_data_1,r_disponible); — 1
149 Enviar_char("00110000",w_data_1,r_disponible); — 0
150 Enviar_char("00110001",w_data_1,r_disponible); — 1
151 Enviar_char("00110000",w_data_1,r_disponible); — 0
152 Enviar_char("00110000",w_data_1,r_disponible); — 0
153 Enviar_char("00110001",w_data_1,r_disponible); — 1
154 Enviar_char("00110001",w_data_1,r_disponible); — 1
155 Enviar_char("00111110",w_data_1,r_disponible); — >
156 —wait for 5 * TbPeriod;
157 r_disponible <= '0';
158
159 wait for 1000 * TbPeriod;
160
161 —wait for 100 * TbPeriod;
162 —TbSimEnded <= '1';
163 end process;
164
165 end tb;

```

ALGORITMO 4.7: Testbench del módulo selector

#### 4.8.2. Resultados obtenidos

En la figura 4.9 se visualiza el comportamiento del sistema con el switch en estado alto. Se puede comprobar que ambas señales llegan al módulo: w\_data\_1 y wr\_uart\_1 son la señal con información y el tren de pulsos que corresponden a no procesar la señal; mientras que w\_data\_2 y wr\_uart\_2 son las mismas pero luego de haber sido procesadas por el sistema de enclavamiento.

4.8. Validación del selector

71

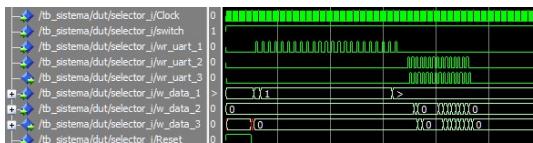


FIGURA 4.9: Simulación del selector.

Las señales w\_data\_3 y wr\_uart\_3 son la salida del bloque selector, que se pue-  
de apreciar copian exactamente a w\_data\_2 y wr\_uart\_2. Cuando el switch se  
encuentra en estado bajo la salida copiará a w\_data\_1 y wr\_uart\_1.

4.8. Validación del selector

73

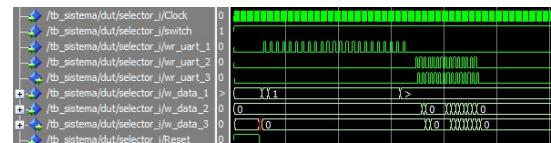


FIGURA 4.9: Simulación del selector.

Las señales w\_data\_3 y wr\_uart\_3 son la salida del bloque selector, que se pue-  
de apreciar copian exactamente a w\_data\_2 y wr\_uart\_2. Cuando el switch se  
encuentra en estado bajo la salida copiará a w\_data\_1 y wr\_uart\_1.

## Capítulo 5

### Conclusiones

En este capítulo se presentan las conclusiones obtenidas, los logros alcanzados y las dificultades encontradas, además de un resumen de los elementos del proyecto que no se incluyeron en la memoria.

#### 5.1. Resultados obtenidos

A lo largo de todo el proyecto se ha visitado una gran cantidad de lugares claves del ámbito ferroviario (estaciones, talleres, oficinas, áreas de pruebas, etc.); se ha conocido a decenas de profesionales del área que, con sus conocimientos y experiencias en el área, han sabido aportar una pieza para este rompecabezas. Ello, combinando tanto el esfuerzo propio como la ayuda de los integrantes del CONICET-GICSAFe, permitió implementar de forma exitosa un prototipo del sistema de enclavamiento electrónico.

El proyecto comenzó como una idea abstracta y ambiciosa que fue evolucionando y adquiriendo mayor definición conforme avanzaban las entrevistas con los diversos actores que en el día a día comandan las instalaciones ferroviarias. Al ir investigando, se iban encontrando nuevas ramificaciones para el proyecto y nuevos enfoques para abordarlo, lo cual elevó su complejidad a tal punto de calificar para una beca de doctorado que iniciará el corriente año.

Durante la Especialización, las modificaciones de metas y alcances fueron una constante, al igual que los cambios de locación. Por lo tanto, el haber encarado la segunda etapa del proyecto en la maestría con un enfoque general, en vistas de automatizar la totalidad del proceso, fue un enorme esfuerzo extra que no estaba contemplado pero cuya necesidad surgió del contexto adverso en el que se desarrolló el sistema. Al final del trabajo, todas esas horas dedicadas a la automatización tuvieron su impacto positivo en cada una de sus aristas.

Los conocimientos en el área de las FPGAs sirvieron enormemente para materializar los diseños planteados, aunque fueron necesarias muchas horas de ensayos y práctica hasta tener no solo un sistema seguro y funcional, sino también uno escalable al automatizar la generación del código.

El haber trabajado en conjunto en otros proyectos de CONICET-GICSAFe a la par del desarrollo del sistema de enclavamientos permitió tomar conciencia de su importancia real como sistema crítico y abre las puertas a una tercera etapa próxima a iniciarse durante 2020 en el Doctorado en Ingeniería de la Universidad de Buenos Aires. Con una base sólida de conocimientos, expectativas mucho más altas y vínculos humanos y profesionales, pero con la confianza y el respaldo que otorgan los avances concretados en esta etapa que llega a su fin.

## Capítulo 5

### Conclusiones

En este capítulo se presentan las conclusiones obtenidas, los logros alcanzados y las dificultades encontradas, además de un resumen de los elementos del proyecto que no se incluyeron en la memoria.

#### 5.1. Resultados obtenidos

A lo largo de todo el proyecto se ha visitado una gran cantidad de lugares claves del ámbito ferroviario (estaciones, talleres, oficinas, áreas de pruebas, etc.); se ha conocido a decenas de profesionales del área que, con sus conocimientos y experiencias en el área, han sabido aportar una pieza para este rompecabezas. Ello, combinando tanto el esfuerzo propio como la ayuda de los integrantes del CONICET-GICSAFe, permitió implementar de forma exitosa un prototipo del sistema de enclavamiento electrónico.

El proyecto comenzó como una idea abstracta y ambiciosa que fue evolucionando y adquiriendo mayor definición conforme avanzaban las entrevistas con los diversos actores que en el día a día comandan las instalaciones ferroviarias. Al ir investigando, se iban encontrando nuevas ramificaciones para el proyecto y nuevos enfoques para abordarlo, lo cual elevó su complejidad a tal punto de calificar para una beca de doctorado que iniciará el corriente año.

Durante la Especialización, las modificaciones de metas y alcances fueron una constante, al igual que los cambios de locación. Por lo tanto, el haber encarado la segunda etapa del proyecto en la maestría con un enfoque general, en vistas de automatizar la totalidad del proceso, fue un enorme esfuerzo extra que no estaba contemplado pero cuya necesidad surgió del contexto adverso en el que se desarrolló el sistema. Al final del trabajo, todas esas horas dedicadas a la automatización tuvieron su impacto positivo en cada una de sus aristas.

Los conocimientos en el área de las FPGAs sirvieron enormemente para materializar los diseños planteados, aunque fueron necesarias muchas horas de ensayos y práctica hasta tener no solo un sistema seguro y funcional, sino también uno escalable al automatizar la generación del código.

El haber trabajado en conjunto en otros proyectos de CONICET-GICSAFe a la par del desarrollo del sistema de enclavamientos permitió tomar conciencia de su importancia real como sistema crítico y abre las puertas a una tercera etapa próxima a iniciarse durante 2020 en el Doctorado en Ingeniería de la Universidad de Buenos Aires. Con una base sólida de conocimientos, expectativas mucho más altas y vínculos humanos y profesionales, pero con la confianza y el respaldo que otorgan los avances concretados en esta etapa que llega a su fin.

En el transcurso del proyecto se han alcanzado los siguientes logros:

- Diseño e implementación de un algoritmo analizador de redes ferroviarias, por lo que pueden analizarse la mayoría de las topologías de las redes ferroviarias argentinas.
- Diseño e implementación de un generador de código en VHDL basado en un grafo ferroviario, por lo que pueden implementarse enclavamientos electrónicos cualquiera sea la locación.
- Diseño e implementación de un generador de tramas de prueba para comandar la plataforma FPGA desde Python, que facilita el testeo de los sistemas generados.
- Publicación de artículos en IEEE Latin America y el Congreso Argentino de Sistemas Embebidos 2019.
- Beca de doctorado en desarrollo estratégico de CONICET 2020-2025.
- Desarrollo del primer sistema ferroviario crítico en el marco del CONICET-

En el transcurso del proyecto se han alcanzado los siguientes logros:

- Diseño e implementación de un algoritmo analizador de redes ferroviarias, por lo que pueden analizarse la mayoría de las topologías de las redes ferroviarias argentinas.
- Diseño e implementación de un generador de código en VHDL basado en un grafo ferroviario, por lo que pueden implementarse enclavamientos electrónicos cualquiera sea la locación.
- Diseño e implementación de un generador de tramas de prueba para comandar la plataforma FPGA desde Python, que facilita el testeo de los sistemas generados.
- Publicación de artículos en IEEE Latin America y el Congreso Argentino de Sistemas Embebidos 2019.
- Beca de doctorado en desarrollo estratégico de CONICET 2020-2025.
- Desarrollo del primer sistema ferroviario crítico en el marco del CONICET-

GICSAFe.

## 5.2. Próximos pasos

La planificación del proyecto contempló desde un inicio que sería un trabajo de dos años a ser realizado en conjunto entre la Especialización y la Maestría de Sistemas Embibidos. Pero el mencionado incremento en la complejidad del sistema abrió las puertas a continuar el proyecto durante el doctorado. Por lo tanto, se mencionan a continuación los pasos a seguir para el próximo tramo del proyecto.

- Optimización del analizador de grafos ferroviarios.
- Integración con la interfaz gráfica desarrollada en UTN-Haedo.
- Realización de pruebas en paralelo con la estación Olivos, gracias al desarrollo del sistema modular del Mg. Ing. Lucas Dórdolo.
- Culminación del generador de test para COCOTB.
- Culminación del generador de tablas de enclavamiento.
- Corrección del funcionamiento de las barreras.
- Elaboración de un análisis formal del sistema de enclavamientos para redes ferroviarias arbitrarias.
- Aplicación de técnicas de redundancia por votación para aumentar la seguridad del sistema.
- Obtención de los parámetros RAMS alcanzados para obtener el grado SIL necesario.

75

GICSAFe.

## 5.2. Próximos pasos

La planificación del proyecto contempló desde un inicio que sería un trabajo de dos años a ser realizado en conjunto entre la Especialización y la Maestría de Sistemas Embibidos. Pero el mencionado incremento en la complejidad del sistema abrió las puertas a continuar el proyecto durante el doctorado. Por lo tanto, se mencionan a continuación los pasos a seguir para el próximo tramo del proyecto.

- Optimización del analizador de grafos ferroviarios.
- Integración con la interfaz gráfica desarrollada en UTN-Haedo.
- Realización de pruebas en paralelo con la estación Olivos, gracias al desarrollo del sistema modular del Mg. Ing. Lucas Dórdolo.
- Culminación del generador de test para COCOTB.
- Culminación del generador de tablas de enclavamiento.
- Corrección del funcionamiento de las barreras.
- Elaboración de un análisis formal del sistema de enclavamientos para redes ferroviarias arbitrarias.
- Aplicación de técnicas de redundancia por votación para aumentar la seguridad del sistema.
- Obtención de los parámetros RAMS alcanzados para obtener el grado SIL necesario.

77

## Bibliografía

- [1] SIEMENS. Siemens proveerá la señalización de una nueva línea ferroviaria principal en Argelia. Disponible: 2018-11-28. 2011. URL: <http://www.aan.siemens.com>.
- [2] Asociación Española de Normalización (UNE). UNE-EN 50126-1:2018. Disponible: 2018-09-12. 2018. URL: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma>.
- [3] Asociación Española de Normalización (UNE). UNE-EN 50128:2012. Disponible: 2012-03-21. 2012. URL: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma>.
- [4] Asociación Española de Normalización (UNE). UNE-EN 50129:2005. Disponible: 2005-03-16. 2005. URL: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma>.
- [5] CONICET-GICSAFE. Grupo de Investigación y Control para la Seguridad y Aplicaciones Ferroviarias. Disponible: 2018-11-28. 2018. URL: <http://www.conicet-gicsafe.com.ar>.
- [6] RITO. Reglamento Interno Técnico Operativo. Disponible: 2018-11-28. 1993. URL: <https://www.argentina.gob.ar/sites/default/files/rito.pdf>.
- [7] Radek Dobias, Hana Kubatova. FPGA based design of the railway interlocking equipment. Ed. por Department of Computer Science y Czech Technical University Prague Engineering. 2004.

## Bibliografía

- [1] I.G. Fiad, *Inversiones para la transformación en ferrocarriles*, Disponible: 2020-04-05. 2019. dirección: [https://www.argentina.gob.ar/sites/default/files/2017/10/18/\\_presentacion\amcham.pdf](https://www.argentina.gob.ar/sites/default/files/2017/10/18/_presentacion\amcham.pdf).
- [2] —, *Avances del plan en infraestructura ferroviaria y su impacto en el desarrollo del país*, Disponible: 2020-04-05. 2019. dirección: [https://www.argentina.gob.ar/sites/default/files/2017/07/24\\\_presentacion\g\\_fiad\pdf..pdf](https://www.argentina.gob.ar/sites/default/files/2017/07/24\_presentacion\g_fiad\pdf..pdf).
- [3] SIEMENS. Siemens proveerá la señalización de una nueva línea ferroviaria principal en Argelia. Disponible: 2018-11-28. 2011. dirección: <http://www.aan.siemens.com>.
- [4] Siemens, *Trackguard Westrace Mk II*. (2014) *Flexible safety processor*. Disponible: 2020-04-05. 2019. dirección: <https://www.mobility.siemens.com/mobility/global/SiteCollectionDocuments/en/rail-solutions/rail-automation/electronic-interlockings/trackguard-westrace-mk2-en.pdf>.
- [5] —, *Trackguard WESTLOCK. Cost-effective, highly available and reliable*. Disponible: 2020-04-05. 2013. dirección: <https://www.mobility.siemens.com/mobility/global/SiteCollectionDocuments/en/rail-solutions/rail-automation/electronic-interlockings/trackguard-westlock-en.pdf>.
- [6] Alstoms, *Alstoms Smartlock equipment at the heart of modular signalling project in the UK*. Disponible: 2020-04-05. 2011. dirección: <https://www.alstom.com/press-releases-news/2011/3/Alstom-Smartlock>.
- [7] Hitachi, *Computer based interlocking*. Disponible: 2019-11-27. 2019. dirección: <http://sts.hitachirail.com/en/products-services/business-segments/computer-based-interlocking>.
- [8] AZD Praha, *Systém pro kolejovou dopravu*, E. S. T. E. 33, ed., Disponible: 2020-04-05. 2018. dirección: <https://www.azd.cz/admin-data/storage/get/198>.
- [9] Thales, *Rout Control Systems*, L. E. L90, ed., Disponible: 2020-04-05. 2019. dirección: <https://www.thalesgroup.com/en/route-control-systems>.
- [10] Bombardier Transportation, *Rail Control Solutions (RCS)*, Disponible: 2020-04-05. 2015. dirección: [http://www.irse.org/knowledge/publicdocuments/\Bombardier \%2001\%20IRSE\\\_BT\%RCS\%Introduction\\_\%20November\%202015\%20final.pdf](http://www.irse.org/knowledge/publicdocuments/\Bombardier \%2001\%20IRSE\_BT\%RCS\%Introduction_\%20November\%202015\%20final.pdf).
- [11] Kyosan, *Railway Signaling Solutions. Interlocking Equipment*, Disponible: 2020-04-05. 2019. dirección: <https://www.kyosan.co.jp/english/product/signal03.html>.
- [12] —, *Kyosan history*, Disponible: 2020-04-05. 2019. dirección: <https://www.kyosan.co.jp/english/company/history02.html>.
- [13] Hima, *COTS Rail Applications*, Disponible: 2020-04-05. 2019. dirección: <https://www.hima.com/en/industries-solutions/cots-rail-references>.
- [14] Asociación Española de Normalización (UNE). UNE-EN 50126-1:2018. Disponible: 2018-09-12. 2018. dirección: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma>.

78

## Bibliografía

- [15] —, UNE-EN 50128:2012, Disponible: 2012-03-21. 2012. dirección: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma>.
- [16] —, UNE-EN 50129:2005, Disponible: 2005-03-16. 2005. dirección: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma>.
- [17] CONICET-GICSAFE. Grupo de Investigación y Control para la Seguridad y Aplicaciones Ferroviarias. Disponible: 2018-11-28. 2018. dirección: <http://www.conicet-gicsafe.com.ar>.
- [18] Trenes Argentinos, Disponible: 2020-04-05. 2019. dirección: <https://www.argentina.gob.ar/transporte/trenes-argentinos>.
- [19] U. Yıldırım, M. S. Durmus, y M. T. Soylemez, *Automatic Interlocking Table Generation for Railway Stations using Symbolic Algebra*. n. IFAC Proceedings Volumes vol 45, ed., Disponible: 2020-04-05. 2012. dirección: <http://dx.doi.org/10.3182/20120912-3-BG-2031.00035>.
- [20] M. Banci y A. Fantechi, *Geographical Versus Functional Modelling by Statecharts of Interlocking Systems*. v. Electronic Notes in Theoretical

- Computer Science, ed., Disponible: 2020-04-05, 2005. dirección: <http://dx.doi.org/10.1016/j.entcs.2004.08.055>.
- [21] RITO, *Reglamento Interno Técnico Operativo*, Disponible: 2018-11-28, 1993. dirección: <https://www.argentina.gob.ar/sites/default/files/rito.pdf>.
- [22] R. Dobias and H. Kubatova, *FPGA based design of the railways interlocking equipments*, E. S. on Digital System Design), ed., 2004.
- [23] Radek Dobias, Hana Kubatova, *FPGA based design of the railway interlocking equipment*, D. of Computer Science y C. T. U. P. Engineering, eds., 2004.
- [24] Altera Corporation, *8 Reasons to Use FPGAs in IEC 61508 Functional Safety Applications*. Altera., ed., 2013.
- [25] Xilinx, *Triple Module Redundancy Design Techniques for Virtex FPGAs*. V. Series, ed., 2006.
- [26] F. L. Kastensmidt, L. Sterpone, L. Carro, y M. S. Reorda, *On the Optimal Design of Triple Modular Redundancy Logic for SRAM-based FPGAs*. Automation y T. in Europe, eds., Disponible: 2020-04-05, 2005. dirección: <http://dx.doi.org/10.1109/DATC.2005.229>.
- [27] Digilent inc., *Arty Z7 Reference Manual*. Disponible: 2020-04-05, 2018. dirección: <https://reference.digilentinc.com/\reference\programmable-logic\arty-z7\reference-manual>.
- [28] Michele Banci,Alessandro Fantechi,Stefania Gnesi, *Some experiences on Formal specification of Railway Interlocking Systems using Statecharts*, 2005.
- [29] Radek Dobias, Hana Kubatova, *FPGA Based Desing of the Railway's Interlocking Equipments*, 10.1109/DSD.2004.1333312, 2004.
- [30] Michele Banci, Alessandro Fantechi, *Geographical Versus Functional Modelling by Statecharts of Interlocking Systems*, 10.1016/j.entcs.2004.08.055, 2005.
- [31] Kirsten Winter, Neil J. Robinson, *Modelling Large Railway Interlockings and Model Checking Small Ones*, 2003.
- [32] Inge Bethke, *Selecting sets of disjoint paths in a railway graph*, 2005.
- [33] Dong Wang,Xiangxian Chen,Hai Huang, *A graph theory based new approach to route location in railway interlocking*, 10.1016/j.cie.2013.09.019, 2013.
- [34] Wai Wong, *A Simple Graph Theory and Its Application in Railway Signalling*.
- [35] Muhammet Ali Nur Oz, Ibrahim Sener, Ozgur Turay Kaymakci, Ilker Ustoglu, Galip Cansever, *Topology Based Automatic Formal Model Generation for Point Automation Systems*, 10.5755/j01.itc.44.1.7382, 2015.