

Modelado e implementación automática de sistemas críticos de enclavamiento ferroviario basados en FPGA

Mg. Ing. Martín Nicolás Menéndez

Tesis doctoral

Director:
Dr. Ing. Ariel Lutenberg (FIUBA-CONICET)

Jurados:
JURADO NUMERO1 (FILIACION)
JURADO NUMERO2 (FILIACION)
JURADO NUMERO3 (FILIACION)

Resumen

Los sistemas de enclavamientos ferroviarios controlan el señalamiento de forma tal de garantizar que las formaciones se desplacen de forma segura, sin colisiones ni descarrilamientos. El señalamiento incluye los semáforos (o señales) que otorgan autoridad a los maquinistas para transitar por las vías, en función del estado de la infraestructura ferroviaria implicada, como pasos a nivel, desvíos, etc. El diseño del señalamiento es un proceso complejo que involucra el análisis de la red ferroviaria, la detección de zonas riesgosas y la correcta ubicación de las señales. La generación automática de la señalización es valiosa y útil cuando se desarrolla una nueva red ferroviaria o cuando se reactiva una red ferroviaria abandonada.

En este contexto, se diseñó un conjunto de herramientas que, a partir del trazado ferroviario, realizan de forma automática el diseño e implementación del señalamiento utilizando un lenguaje de descripción de hardware. La etapa encargada del diseño del señalamiento ferroviario es el Analizador de Redes Ferroviarias (RNA, por sus siglas en inglés). En tanto que de la implementación en VHDL se encarga el Generador Automático de Código (ACG, por sus siglas en inglés). Ambas herramientas intercambian información entre sí y con los usuarios, mediante el estándar abierto de intercambio de datos de infraestructura ferroviaria railML. Finalmente, el Generador Automático de Interfaz Gráfica (AGG, por sus siglas en inglés), construye una interfaz interactiva para el operador, que permite visualizar el estado del enclavamiento y comandarlo en tiempo real.

El señalamiento generado incluye el número de las señales necesarias, su posición y orientación, además de la tabla de enclavamiento. La tabla de enclavamientos, el cumplimiento de los principios de diseño ferroviario y la sintaxis del archivo railML generado son validados por el mismo RNA. El archivo railML generado junto con el modelo de comportamiento dinámico, definido en redes de Petri, es utilizado por el ACG para implementar el sistema de enclavamiento. La interfaz gráfica generada a medida de cada sistema por el AGG permite interactuar con el sistema implementado.

Abstract

Railway interlocking systems controls signaling to guarantee that trains operate safely, avoiding collisions and derailments.

Signaling includes the traffic lights (or signals) that authorize train drivers to transit over the next railway tracks, based on the state of the associated railway infrastructure, such as level crossings, switches, etc. Signaling design is a complex process that involves railway network analysis, detection of zones where collisions or derailments are likely, and accurately positioning the signals. The automatic generation of signalling is particularly valuable and beneficial when developing a new railway network or reactivating an abandoned railway network.

In this context, a set of tools was designed to automatically perform the design and implementation of railway signaling using a hardware description language, based on the railway layout. The stage responsible for the design of railway signaling is the Railway Network Analyzer (RNA). The implementation in VHDL is handled by the Automatic Code Generator (ACG). Both tools exchange information with each other and with users through the open standard for railway infrastructure data exchange, railML. Finally, the Automatic Graphical User Interface Generator (AGG) constructs an interactive interface for the operator, allowing real-time visualization and control of the interlocking status.

The generated signaling includes the number of necessary signals, their position and orientation, as well as the interlocking table. The interlocking table, compliance with railway design principles, and the syntax of the generated railML file are validated by the RNA itself. The generated railML file, along with the dynamic behavior model defined in Petri nets, is used by the ACG to implement the interlocking system. The graphical interface custom-generated for each system by the AGG allows interaction with the implemented system.

Agradecimientos

A lo largo de mi vida he conocido muchas personas que me apoyaron, guiaron, influenciaron e inspiraron. No me alcanzan las palabras para agradecerles a todos ellos, pero trataré de, por medio de esta dedicatoria, darles su merecido reconocimiento.

A mis padres, Roberto y Silvina, por su infinito amor, su apoyo y su confianza en mi. A mis hermanos, Agustín y Sofía, por su cariño, por su amistad y por ser los mejores hermanos. A mis sobrinas, Valentina y Belén, por sus sonrisas, que deseo que siempre tengan. A mis abuelos, por motivarme a leer, aprender, construir, romper, inventar. A mi gata Mishu, por acompañarme desde el primer circuito.

A mi director y mentor, Ariel Lutenberg, el mejor docente que tuve la suerte de encontrar en la FIUBA, por todas las oportunidades brindadas, por ayudarme a cumplir este sueño. A mis pasados directores y profesores que me acompañaron a lo largo de este proyecto: Pablo Gómez, Facundo Larosa y Nicolás Álvarez. A mi compañero de laboratorio, Santiago Germino, que siempre me aconseja cómo mejorar en cada artículo. A cada miembro de GICSAFe por todo el trabajo realizado a lo largo de los años. A cada maestro que tuve en la escuela desde el jardín de infantes hasta el secundario, cada uno contribuyó con su granito de arena a mi amor por las matemáticas, la ciencia, la investigación y por incentivar mi curiosidad.

Es mi deseo que esta investigación contribuya a la ingeniería, a la ciencia, y a que nuestra sociedad tenga un sistema ferroviario mas robusto y seguro.

«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)

Índice general

1. Introducción	1
1.1. Topologías ferroviarias	1
1.1.1. Derivación ferroviaria	2
1.1.2. Estación de baja densidad	2
1.1.3. Estación de alta densidad	3
1.1.4. Estación Terminal	4
1.2. Principios de señalamiento ferroviario	5
1.3. Sistemas de enclavamiento	6
1.3.1. Tipos de sistemas de enclavamiento	6
1.3.2. Tablas de enclavamientos	9
1.3.3. Enfoque funcional	11
1.3.4. Enfoque geográfico	12
1.4. Estándares de seguridad	12
1.4.1. Estándares EN-50126, EN-50128 y EN-50129	12
1.4.2. Redundancia y diversidad en sistemas ferroviarios	14
1.5. RailTopoModel y RailML	16
1.5.1. RailTopoModel	16
1.5.1.1. Paquete base	16
1.5.1.2. Modelo de grafos	17
1.5.1.3. Topología y niveles	17
1.5.1.4. Posicionamiento	19
1.5.2. RailML 3.0	19
1.5.3. Uso del estándar railML en la industria ferroviaria	20
1.6. Estado del arte	20
1.6.1. Empresas del sector ferroviario	20
1.6.2. Antecedentes de trabajos académicos en la temática	21
1.6.3. GICSAFe y los trabajos realizados	21
1.7. Herramienta propuesta	23
1.7.1. Enfoque aplicado	23
1.7.2. Arquitectura del sistema	24
1.7.3. Amenazas a la validez del sistema	25
2. Railway Network Analyzer (RNA)	27
2.1. Biblioteca RailML	27
2.1.1. Clase <i>Metadata</i>	28
2.1.2. Clase <i>Common</i>	29
2.1.3. Clase <i>Infraestructure</i> y redes de grafos	30
2.1.4. Vías	33
2.1.5. Fin de vía y transiciones	36

2.1.6. Sistemas de detección de formaciones ferroviarias	37
2.1.7. Estaciones ferroviarias	39
2.1.8. Cruces ferroviarios	41
2.1.9. Cambios de vías	42
2.1.10. Señales ferroviarias	50
2.2. Algoritmos de generación de señalamiento	52
2.2.1. Señalización en fin de vía y transiciones	53
2.2.2. Detectores de posición y velocidad.	54
2.2.3. Plataformas ferroviarias	56
2.2.4. Cruces ferroviarios	57
2.2.5. Cambios de vías	58
2.3. Algoritmos de simplificación de señalamiento	60
2.3.1. Algoritmo de simplificación por herencia vertical	60
2.3.2. Algoritmo de simplificación por herencia horizontal	64
2.3.3. Limpieza de señales	66
2.4. Generación de las tablas de enclavamiento	67
2.5. Validación del señalamiento y las tablas de enclavamiento	69
2.6. Comprobación de principios de señalamiento ferroviario	71
2.6.1. Comprobación del principio de autoridad	71
2.6.2. Comprobación del principio de claridad	72
2.6.3. Comprobación del principio de anticipación	72
2.6.4. Comprobación del principio de granularidad	73
2.6.5. Comprobación del principio de terminalidad	74
2.6.6. Comprobación del principio de infraestructura	75
2.6.7. Comprobación del principio de no bloqueo	75
3. Automatic Code Generator (ACG)	77
3.1. Sistemas de enclavamiento	77
3.1.1. Bloqueo de máquina de cambios por ocupación	77
3.1.2. Requerimiento de rutas y bloqueo de cambios en ruta	78
3.1.3. Protección por aproximación en cancelación de ruta	79
3.1.4. Protección por solape	80
3.1.5. Doble recubrimiento	80
3.1.6. Liberación secuencial	81
3.2. Arquitectura del sistema	82
3.2.1. Modulo UART	85
3.2.2. Módulo Detector	87
3.2.3. Módulo Decoder	89
3.2.4. Módulo Encoder	91
3.2.5. Módulo Printer	92
3.2.6. Módulo Selector	93
3.2.7. Modulo Network	94
3.2.8. Módulo genérico de las barreras ferroviarias	98
3.2.9. Módulo genérico de los cambios de vías simples	101
3.2.10. Módulo genérico de los cambios de vías dobles	102
3.2.11. Módulo genérico de los cambios en tijeras	105
3.2.12. Módulo genérico de los netElements	106
3.2.13. Módulo genérico de las señales ferroviarias	108
3.2.14. Módulo genérico de las rutas ferroviarias	110
3.3. Redundancia	112
3.4. Plataforma utilizada	112

4. Resultados obtenidos	114
4.1. Ejemplo 1	114
4.1.1. Topología ferroviaria original	114
4.1.2. Señalamiento original	115
4.1.3. Generación de señalamiento paso a paso	116
4.1.4. Red de grafos generada por el RNA	126
4.1.5. Señalamiento generado por el RNA	126
4.1.6. Validación del sistema de enclavamientos	127
4.1.7. Sistema generado por el ACG	129
4.1.8. Interfaz gráfica de usuario	135
4.1.9. Comparación entre todos los ejemplos	139
5. Conclusiones	143
5.1. Resultados obtenidos	143
5.2. Trabajo futuro	144
A. Ejemplo 2	145
A.1. Topología ferroviaria original	145
A.2. Señalamiento original	145
A.3. Generación de señalamiento paso a paso	146
A.4. Red de grafos generada por el RNA	152
A.5. Señalamiento generado por el RNA	153
A.6. Validación del sistema de enclavamientos	154
A.7. Sistema generado por el ACG	155
B. Ejemplo 3	156
B.1. Topología ferroviaria original	156
B.2. Señalamiento original	156
B.3. Generación de señalamiento paso a paso	159
B.4. Red de grafos generada por el RNA	179
B.5. Señalamiento generado por el RNA	180
B.6. Validación del sistema	184
B.7. Sistema generado por el ACG	185
C. Ejemplo 4	187
C.1. Topología ferroviaria original	187
C.2. Señalamiento original	187
C.3. Generación de señalamiento paso a paso	191
C.4. Red de grafos generada por el RNA	212
C.5. Señalamiento generado por el RNA	213
C.6. Validación del sistema de enclavamientos	217
C.7. Sistema generado por el ACG	221
D. Ejemplo 5	222
D.1. Topología ferroviaria original	222
D.2. Señalamiento original	222
D.3. Generación de señalamiento paso a paso	224
D.4. Red de grafos generada por el RNA	233
D.5. Señalamiento generado por el RNA	233
D.6. Validación del sistema	233
D.7. Sistema generado por el ACG	234

E. Ejemplo 6	237
E.1. Topología ferroviaria original	237
E.2. Señalamiento original	237
E.3. Generación de señalamiento paso a paso	238
E.4. Red de grafos generada por el RNA	248
E.5. Señalamiento generado por el RNA	248
E.6. Validación del sistema	249
E.7. Sistema generado por el ACG	251
F. Ejemplo 7	253
F.1. Topología ferroviaria original	253
F.2. Señalamiento original	254
F.3. Generación de señalamiento paso a paso	255
F.4. Red de grafos generada por el RNA	264
F.5. Señalamiento generado por el RNA	265
F.6. Validación del sistema	266
F.7. Sistema generado por el ACG	267
G. Ejemplo 8	268
G.1. Topología ferroviaria original	268
G.2. Señalamiento original	268
G.3. Generación de señalamiento paso a paso	269
G.4. Red de grafos generada por el RNA	277
G.5. Señalamiento generado por el RNA	277
G.6. Validación del sistema	278
G.7. Sistema generado por el ACG	279
H. Ejemplo 9	281
H.1. Topología ferroviaria original	281
H.2. Señalamiento original	281
H.3. Generación de señalamiento paso a paso	282
H.4. Red de grafos generada por el RNA	292
H.5. Señalamiento generado por el RNA	292
H.6. Validación del sistema de enclavamientos	293
H.7. Sistema generado por el ACG	294

Índice de Figuras

1.1.	Topología de derivación ferroviaria	2
1.2.	Topología de estación de baja densidad	3
1.3.	Topología de Estación de alta densidad	4
1.4.	Topología terminal	5
1.5.	Sistema de enclavamiento mecánico basado en palancas.	7
1.6.	Sistema de enclavamiento electromecánico basado en relés.	8
1.7.	Panel de control central de un sistema de enclavamientos.	9
1.8.	Ejemplo de asignación de rutas de izquierda a derecha (arriba) y de derecha a izquierda (abajo).	10
1.9.	Estándar IEC 61508.	13
1.10.	Sistema con redundancia 2oo3 y diversidad.	15
1.11.	Alcance del modelo de UIC RailTopoModel.	16
1.12.	Modelado en grafos de un cambio de vías simple.	17
1.13.	Estructura de capas de RailTopoModel.	18
1.14.	Niveles microscópico, mesoscópico y macroscópico.	18
1.15.	Trabajos realizados e hitos del proyecto.	22
1.16.	Flujo de trabajo de la presente tesis.	23
1.17.	Arquitectura del sistema	24
2.1.	Distintos tipos de redes de grafos.	32
2.2.	Vías ferroviarias y trocha.	34
2.3.	Vías ascendentes y descendentes.	35
2.4.	Topología de ejemplo con finales de vía relativos y absolutos.	36
2.5.	Círculo de vía (01) y contador de ejes (AxC01).	37
2.6.	Círculo de vía libre y ocupado.	38
2.7.	Contadores de ejes.	39
2.8.	Estación de doble plataforma.	40
2.9.	Cruces de vías vehiculares y peatonales.	41
2.10.	Circulaciones posibles en cambios de vías.	43
2.11.	Máquina de cambios.	43
2.12.	Cambio de vías de estación Matheu, Linea Mitre.	44
2.13.	Comando, indicación y correspondencia en un cambio de vías.	44
2.14.	Aspectos de cada semáforo y múltiples señales en un semáforo.	51
2.15.	Ejemplo de asignación de señalización ramificada.	52
2.16.	Señalamiento generado para finales de vía relativos y absolutos.	54
2.17.	Señalamiento generado para contadores de ejes y circuitos de vías.	56
2.18.	Señalamiento generado para estaciones ferroviarias.	57
2.19.	Señalamiento generado para cruces ferroviarios.	58
2.20.	Señalamiento generado para cambios de vías.	59
2.21.	Señalamiento generado, simplificado por algoritmo de herencia vertical.	61

2.22. Señalamiento generado, simplificado por algoritmo de herencia vertical.	63
2.23. Señalamiento generado, simplificado por algoritmo de herencia horizontal.	65
2.24. Signalling simplified in Example 1 with three routes indicated.	68
 3.1. Bloqueo de máquina de cambios por ocupación de secciones adyacentes.	78
3.2. Bloqueo de rutas conflictivas.	78
3.3. Formación aproximándose al inicio de la ruta.	79
3.4. La ruta es cancelada mientras la formación se approxima.	79
3.5. La formación se detiene exitosamente previo a la señal de peligro.	79
3.6. La formación no se detiene previo a la señal de peligro.	80
3.7. Formación ignora señal a peligro y se activa la protección por solape.	80
3.8. Protección por doble recubrimiento.	81
3.9. Formación iniciando una ruta ferroviaria.	81
3.10. Formación activando el bloqueo por ocupación y el bloqueo por solape.	82
3.11. Liberación secuencial de la infraestructura por detrás de la formación.	82
3.12. Liberación secuencial de la infraestructura por delante de la formación.	82
3.13. Diagrama en bloques de una FSMD genérica.	83
3.14. Arquitectura general del sistema generado por el ACG.	84
3.15. Topología de derivación ferroviaria.	85
3.16. FPGA.	86
3.17. Tramas de datos y paquete de datos.	86
3.18. FSMD del módulo <i>Detector</i>	87
3.19. Diagrama de estados del módulo <i>Detector</i>	88
3.20. FSMD del módulo <i>Decoder</i>	90
3.21. FSMD del módulo <i>Encoder</i>	91
3.22. FSMD del módulo <i>Printer</i>	92
3.23. Diagrama de estados del módulo <i>Printer</i>	93
3.24. FSMD del módulo <i>Selector</i>	94
3.25. FSMD del módulo <i>Network</i>	95
3.26. Red de Petri del modelo dinámico de enclavamiento de elementos ferroviarios genéricos.	97
3.27. FSMD del módulo genérico de <i>LevelCrossing</i>	99
3.28. Red de Petri del modelo dinámico de <i>LevelCrossing</i>	100
3.29. FSMD del módulo genérico de <i>SingleSwitches</i>	101
3.30. Red de Petri del modelo dinámico de <i>SingleSwitches</i>	102
3.31. FSMD del módulo genérico de <i>DoubleSwitches</i>	103
3.32. Red de Petri del modelo dinámico de <i>DoubleSwitches</i>	104
3.33. FSMD del módulo genérico de <i>ScissorCrossings</i>	105
3.34. Red de Petri del modelo dinámico de <i>ScissorCrossings</i>	106
3.35. FSMD del módulo genérico de <i>NetElements</i>	107
3.36. Red de Petri del modelo dinámico de <i>NetElements</i>	108
3.37. FSMD del módulo genérico de <i>Signals</i>	109
3.38. Red de Petri del modelo dinámico de <i>Signals</i>	110
3.39. FSMD del módulo genérico de <i>Routes</i>	111
3.40. Red de Petri del modelo dinámico de <i>Routes</i>	112
3.41. Plataforma FPGA Xilinx Arty Z7-20.	113
 4.1. Topología ferroviaria del ejemplo 1 sin señalamiento.	115
4.2. Señalamiento original del ejemplo 1.	115
4.3. Señalamiento generado por el RNA para proteger el fin de vía.	120
4.4. Señalamiento generado por el RNA para proteger las junturas.	120
4.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	121

4.6. Señalamiento generado por el RNA para proteger los cambios de vías.	121
4.7. Señalamiento generado y simplificado por el RNA.	122
4.8. Red de grafos generada por el RNA para el ejemplo 1.	126
4.9. Archivos generador por el ACG para el ejemplo 1.	130
4.10. Interfaz del entorno de desarrollo Vivado para el ejemplo 1.	134
4.11. Interfaz gráfica del ejemplo 1.	136
4.12. Nuevo señalamiento con ne22 ocupado.	137
4.13. Nuevo señalamiento con Sw12 en reversa.	137
4.14. Ruta 13 del ejemplo 1 en ejecución.	138
4.15. Ruta 12 del ejemplo 1 en ejecución.	139
4.16. Ruta 19 del ejemplo 1 en ejecución1.	139
4.17. Uso de Look-Up-Tables (LUTs) y Flip-Flops (FFs) en función de N.	141
4.18. Porcentaje de uso de Look-Up-Tables (LUTs) y Flip-Flops (FFs) en función de N, considerando la plataforma Arty Z7 20.	141
A.1. Topología ferroviaria del ejemplo 2 sin señalamiento.	145
A.2. Señalamiento original del ejemplo 2.	146
A.3. Señalamiento generado por el RNA para proteger el fin de vía.	149
A.4. Señalamiento generado por el RNA para proteger las junturas.	149
A.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	149
A.6. Señalamiento generado por el RNA para proteger las máquinas de cambios.	150
A.7. Señalamiento generado y simplificado por el RNA.	150
A.8. Red de grafos generada por el RNA para el ejemplo 2.	153
B.1. Topología ferroviaria del ejemplo 3 sin señalamiento.	156
B.2. Señalamiento original del ejemplo 3.	157
B.3. Señalamiento generado por el RNA para proteger el fin de vía.	167
B.4. Señalamiento generado por el RNA para proteger las junturas.	168
B.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	168
B.6. Señalamiento generado por el RNA para proteger los cambios de vías.	168
B.7. Señalamiento generado y simplificado por el RNA.	170
B.8. Red de grafos generada por el RNA para el ejemplo 3.	180
C.1. Topología ferroviaria del ejemplo 4 sin señalamiento.	187
C.2. Señalamiento original del ejemplo 4.	188
C.3. Señalamiento generado por el RNA para proteger el fin de vía.	200
C.4. Señalamiento generado por el RNA para proteger las junturas.	201
C.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	201
C.6. Señalamiento generado por el RNA para proteger los cambios de vías.	202
C.7. Señalamiento generado y simplificado por el RNA.	203
C.8. Red de grafos generada por el RNA para el ejemplo 4.	213
D.1. Topología ferroviaria del ejemplo 5 sin señalamiento.	222
D.2. Señalamiento original del ejemplo 5.	223
D.3. Señalamiento generado por el RNA para proteger el fin de vía.	227
D.4. Señalamiento generado por el RNA para proteger las junturas.	228
D.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	228
D.6. Señalamiento generado por el RNA para proteger los cambios de vías.	229
D.7. Señalamiento generado y simplificado por el RNA.	230
D.8. Red de grafos generada por el RNA para el ejemplo 5.	233

E.1. Topología ferroviaria del ejemplo 6 sin señalamiento.	237
E.2. Señalamiento original del ejemplo 6.	238
E.3. Señalamiento generado por el RNA para proteger el fin de vía.	242
E.4. Señalamiento generado por el RNA para proteger las junturas.	242
E.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	243
E.6. Señalamiento generado por el RNA para proteger las máquinas de cambios.	243
E.7. Señalamiento generado y simplificado por el RNA.	244
E.8. Red de grafos generada por el RNA para el ejemplo 6.	248
F.1. Topología ferroviaria del ejemplo 7 sin señalamiento.	253
F.2. Señalamiento original del ejemplo 7.	254
F.3. Señalamiento generado por el RNA para proteger el fin de vía.	258
F.4. Señalamiento generado por el RNA para proteger las junturas.	259
F.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	260
F.6. Señalamiento generado por el RNA para proteger los cambios de vías.	261
F.7. Señalamiento generado y simplificado por el RNA.	262
F.8. Red de grafos generada por el RNA para el ejemplo 7.	265
G.1. Topología ferroviaria del ejemplo 8 sin señalamiento.	268
G.2. Señalamiento original del ejemplo 8.	269
G.3. Señalamiento generado por el RNA para proteger el fin de vía.	272
G.4. Señalamiento generado por el RNA para proteger las junturas.	272
G.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	273
G.6. Señalamiento generado por el RNA para proteger los cambios de vías.	273
G.7. Señalamiento generado y simplificado por el RNA.	274
G.8. Red de grafos generada por el RNA para el ejemplo 8.	277
H.1. Topología ferroviaria del ejemplo 9 sin señalamiento.	281
H.2. Señalamiento original del ejemplo 9.	281
H.3. Señalamiento generado por el RNA para proteger el fin de vía.	285
H.4. Señalamiento generado por el RNA para proteger las junturas.	286
H.5. Señalamiento generado por el RNA para proteger plataformas y cruces de vía.	286
H.6. Señalamiento generado por el RNA para proteger los cambios de vías.	286
H.7. Señalamiento generado y simplificado por el RNA.	288
H.8. Red de grafos generada por el RNA para el ejemplo 9.	292

Índice de Tablas

1.1. Tabla de enclavamientos (rutas de izquierda a derecha).	10
1.2. Tabla de enclavamientos (rutas de derecha a izquierda).	10
1.3. Tabla de enclavamientos completa.	11
1.4. SIL en función de la Probabilidad de Fallas/Hora (PFH).	14
3.1. Estados de enclavamiento de cada elemento ferroviario.	95
4.1. Tabla de enclavamiento original del ejemplo 1.	116
4.2. Tabla de enclavamiento del ejemplo 1 generada por el RNA.	127
4.3. Equivalencias entre las rutas originales y las generadas por el RNA.	128
4.4. Síntesis e implementación del ejemplo 1 generado por el ACG.	135
4.5. Comparación entre los ejemplos generados por el ACG.	140
A.1. Tabla de enclavamiento original del ejemplo 2.	146
A.2. Tabla de enclavamiento del ejemplo 2 generada por el RNA.	153
A.3. Equivalencias entre las rutas originales y las generadas por el RNA.	154
A.4. Síntesis e implementación del ejemplo 2 generado por el ACG.	155
B.1. Tabla de enclavamiento original del ejemplo 3.	158
B.2. Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 1 a 15).	181
B.3. Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 16 a 30).	181
B.4. Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 31 a 45).	182
B.5. Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 46 a 60).	182
B.6. Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 61 a 75).	183
B.7. Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 75 a 91).	183
B.8. Equivalencias entre las rutas originales y las generadas por el RNA.	184
B.9. Síntesis e implementación del ejemplo 3 generado por el ACG.	186
C.1. Tabla de enclavamiento original del ejemplo 4 (Rutas 1 a 15).	188
C.2. Tabla de enclavamiento original del ejemplo 4 (Rutas 16 a 30).	189
C.3. Tabla de enclavamiento original del ejemplo 4 (Rutas 31 a 45).	189
C.4. Tabla de enclavamiento original del ejemplo 4 (Rutas 46 a 160).	190
C.5. Tabla de enclavamiento original del ejemplo 4 (Rutas 61 a 76).	191
C.6. Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 1 a 15).	214
C.7. Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 16 a 30).	214
C.8. Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 31 a 45).	215
C.9. Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 46 a 60).	215
C.10. Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 61 a 75).	216
C.11. Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 76 a 89).	216
C.12. Equivalencias entre las rutas 1 a 15 originales y las generadas por el RNA.	217

C.13. Equivalencias entre las rutas 16 a 30 originales y las generadas por el RNA.	218
C.14. Equivalencias entre las rutas 31 a 45 originales y las generadas por el RNA.	218
C.15. Equivalencias entre las rutas 46 a 60 originales y las generadas por el RNA.	219
C.16. Equivalencias entre las rutas 61 a 77 originales y las generadas por el RNA.	220
C.17. Síntesis e implementación del ejemplo 4 generado por el ACG.	221
D.1. Tabla de enclavamiento original del ejemplo 5.	223
D.2. Tabla de enclavamiento del ejemplo 5 generada por el RNA.	234
D.3. Equivalencias entre las rutas originales y las generadas por el RNA.	235
D.4. Síntesis e implementación del ejemplo 5 generado por el ACG.	236
E.1. Tabla de enclavamiento original del ejemplo 6.	238
E.2. Tabla de enclavamiento del ejemplo 6 generada por el RNA.	249
E.3. Equivalencias entre las rutas originales y las generadas por el RNA.	250
E.4. Síntesis e implementación del ejemplo 6 generado por el ACG.	252
F.1. Tabla de enclavamiento original del ejemplo 7.	254
F.2. Tabla de enclavamiento del ejemplo 7 generada por el RNA.	265
F.3. Equivalencias entre las rutas originales y las generadas por el RNA.	266
F.4. Síntesis e implementación del ejemplo 7 generado por el ACG.	267
G.1. Tabla de enclavamiento original del ejemplo 8.	269
G.2. Tabla de enclavamiento del ejemplo 8 generada por el RNA.	277
G.3. Equivalencias entre las rutas originales y las generadas por el RNA.	278
G.4. Síntesis e implementación del ejemplo 8 generado por el ACG.	280
H.1. Tabla de enclavamiento del ejemplo 9 generada por el RNA.	293
H.2. Síntesis e implementación del ejemplo 9 generado por el ACG.	295

Capítulo 1

Introducción

El diseño de los sistemas de señalamiento es un proceso complejo que involucra, principalmente, tres etapas: el análisis de la red ferroviaria, la detección de zonas de mayor probabilidad de descarrilamientos y colisiones, y la óptima localización de las señales correctas para cada función [1-5]. La generación automática del señalamiento es de gran importancia y utilidad para el desarrollo de redes ferroviarias nuevas o para revitalizar redes ferroviarias abandonadas o en desuso. Es relevante incluso en redes ferroviarias que son alteradas debido a la adición, modificación o eliminación de elementos ferroviarios tales como pasos a nivel o plataformas, lo cual implica que el señalamiento debe actualizarse en consecuencia.

En este trabajo presentamos una herramienta capaz de automatizar el diseño del señalamiento y la implementación del sistema de enclavamiento ferroviario para cualquier locación, dado únicamente el diseño de su traza ferroviaria. Se discutirán los detalles del diseño de la herramienta, su arquitectura, casos de uso y aplicaciones en locaciones teóricas y reales.

1.1. Topologías ferroviarias

Las redes ferroviarias presentan dos piezas fundamentales en su infraestructura: su topología y los elementos ferroviarios que la componen. La topología es el entramado de vías férreas, cuyo diseño busca cumplir una función particular, interconectando diversos elementos ferroviarios. Los elementos ferroviarios pueden ser para determinar la ubicación del tren, para delimitar la circulación de vehículos en cruces ferroviarios, permitir el ascenso y descenso de pasajeros, o para modificar dinámicamente los caminos por los que los trenes circulan, entre otras funciones [6-9].

Muchos de los elementos ferroviarios inciden en la circulación del tren, por lo que el conductor debe ser informado de su estado. Es tarea del señalamiento ferroviario alertar a los conductores ferroviarios de cualquier elemento que pueda representar un peligro, evitando así colisiones con otras formaciones o descarrilamientos en zonas críticas [2-5].

El señalamiento ferroviario incluye un elemento fundamental: los semáforos ferroviarios, de ahora en adelante denominados "señales". Las señales informan al conductor de la habilitación o denegación de uso de las vías posteriores mediante su color, denominado aspecto. Cada señal puede presentar un único aspecto por vez de un conjunto posible que varía según el país o la región. Los aspectos utilizados en Argentina son el verde (permitido avanzar), amarillo (atención) y rojo (detenerse) [10]. Algunas señales pueden no tener el aspecto verde (señales de maniobras de dos aspectos) o incorporar un aspecto extra entre el rojo y el amarillo (señales de cuatro aspectos que incluyen el doble amarillo).

Dos señales consecutivas con la misma dirección y sentido constituyen una ruta ferroviaria [10-12]. Los operarios ferroviarios solicitan al sistema de enclavamientos las rutas que necesitan

de acuerdo con la logística deseada. El sistema de enclavamientos habilitará o denegará las rutas solicitadas en función del estado de los elementos ferroviarios cercanos y de las demás rutas activas. Esta función es vital para el sistema ferroviario y su fin último es permitir la circulación de trenes de forma segura o, de no ser posible, no permitir circulación alguna [3, 9, 11, 12].

En las siguientes subsecciones, se describen algunas de las topologías ferroviarias más ampliamente utilizadas.

1.1.1. Derivación ferroviaria

Cuando es necesario conectar dos puntos separados por una distancia de decenas o cientos de kilómetros donde hay poco tráfico ferroviario, resulta económicamente poco conveniente construir vías en ambos sentidos. No obstante, construir una sola vía bidireccional presenta inconvenientes logísticos notorios: una formación que circule entre los puntos A y B excluye a cualquier formación que quiera circular de B a A sin colisionar.

La solución mas utilizada emplea islas de enclavamiento a modo de bypass cada cierta cantidad de kilómetros, como se ilustra en la Figura 1.1. Estas islas permiten que las formaciones puedan cruzarse sin riesgo de colisión. La primer formación en llegar a la isla de enclavamientos accede al bypass por la vía superior y espera a que la formación en sentido contrario circule por la vía inferior. Una vez despejado el camino que resta por recorrer, la formación reingresa a la vía principal y retoma su marcha.

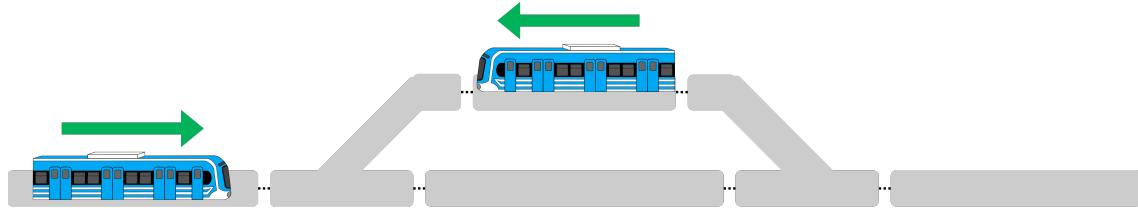


Figura 1.1: Topología de derivación ferroviaria.

Las topologías de derivación ferroviaria se utilizan principalmente para transportar materias primas entre locaciones rurales a grandes distancias de los puestos. Es deseable tanto una logística óptima, para transportar mas bienes y mas rápido, como un sistema seguro que garantice que los bienes lleguen a destino.

1.1.2. Estación de baja densidad

En entornos urbanos donde las estaciones ferroviarias se encuentran separadas entre sí por unos pocos kilómetros es necesaria una interconectividad mayor. Si por la estación ferroviaria solamente pasan uno o dos ramales ferroviarias, una topología de estación de baja densidad como la ilustrada en la Figura 1.2 es una solución al problema planteado [13].



Figura 1.2: Topología de estación de baja densidad.

El cruce entre el trazado ferroviario y el trazado vehicular se denomina paso a nivel. El sistema de enclavamientos deberá garantizar que el paso a nivel se encuentre despejado de vehículos y peatones antes de permitir la circulación de trenes sobre este. Esto se logra mediante el uso de una barrera ferroviaria, que baja el brazo de barrera cuando se detectan formaciones ferroviarias en las proximidades del paso a nivel [14].

Las topologías de estación de baja densidad suelen contar con dos vías unidireccionales en sentido ascendente y descendente. Las vías ascendentes son aquellas por las que las formaciones circulan en la dirección del kilometraje creciente. Mientras que las vías descendentes son aquellas por las que circulan en la dirección del kilometraje decreciente [10]. El kilómetro cero es la estación principal de la línea ferroviaria, como por ejemplo: Plaza Constitución (Línea Roca), Once de Septiembre (Línea Sarmiento) o Retiro (Línea Mitre y Línea San Martín). Para invertir su sentido de circulación las formaciones pueden cambiar de vía ascendente a descendente, o viceversa, utilizando un cambio ferroviario.

1.1.3. Estación de alta densidad

A medida que mas ramales ferroviarios coexisten en la misma línea se vuelve inevitable que varias líneas compartan la misma estación utilizando diferentes plataformas en paralelo. Con una logística mas flexible, las diferentes líneas incluso pueden utilizar de forma alternada las mismas plataformas y, por lo tanto, las mismas vías principales. Además, es necesario contar con mecanismos para retirar trenes de la red para su mantenimiento y volver a inyectarlos a la red cuando la demanda aumente. Esto se logra por medio de talleres ferroviarios en las inmediaciones de las estaciones que actúan como un hub ferroviario. Una topología de estación de alta densidad se ilustra en la Figura 1.3.



Figura 1.3: Topología de Estación de alta densidad.

Las tareas del sistema de enclavamientos van aumentando en complejidad a medida que se suman nuevos elementos ferroviarios. Por ejemplo, debe asegurar que las formaciones circulen con seguridad pero sin afectar la disponibilidad del sistema.

1.1.4. Estación Terminal

Las estaciones terminales presentan una gran cantidad de vías principales y plataformas en paralelo, en las cuales confluyen una o varias líneas ferroviarias. A diferencia de estaciones de alta densidad que pueden presentar finales de vía relativos, las estaciones terminales poseen finales de vía absolutos. Es decir, las formaciones que circulan por la vía descendente deberán detener su marcha completamente antes de llegar al final de vía, para luego retomar su marcha en sentido contrario, por la vía ascendente. Esta operación puede realizarse de manera inmediata en formaciones con locomotoras eléctricas en ambos extremos del tren o con locomotoras diesel luego de varias maniobras que requieren el uso de diversos cambios de vías. En la Figura 1.4 se ilustra un ejemplo de una estación terminal.

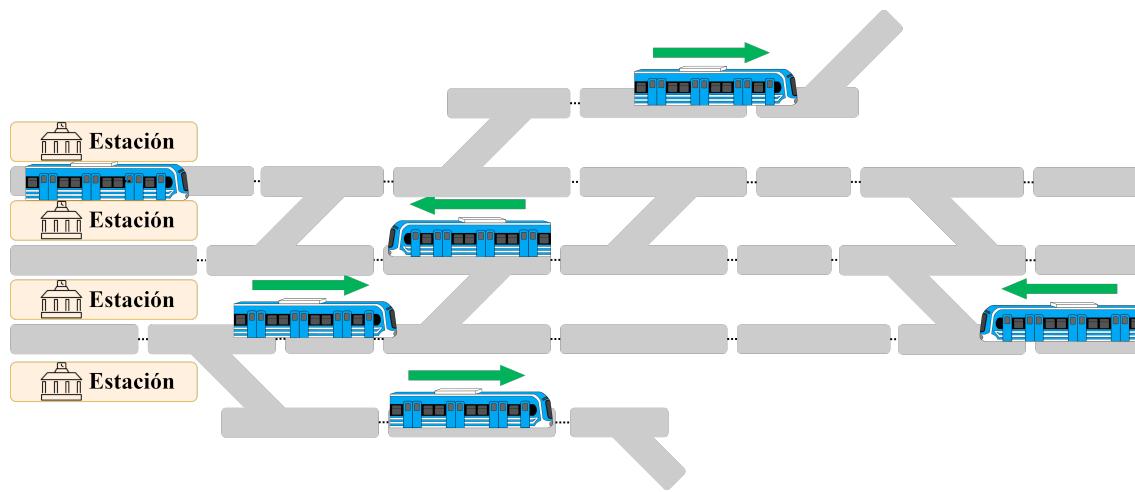


Figura 1.4: Topología terminal.

En las estaciones terminales suelen confluir la información en tiempo real de la terminal y las estaciones más próximas de la línea, o incluso la información en tiempo real de la totalidad de la línea. Esta característica, además de ser la estación de mayor tamaño de la línea, les otorga una jerarquía tal que suelen concentrar parcial o totalmente el control del señalamiento de la red. Las decisiones tomadas en una estación terminal tienen un gran impacto en el sistema de transporte de toda la línea, directa o indirectamente. Estas operaciones deben considerar cientos o miles de estados en simultáneo, por lo que ejecutarlas de forma manual es muy complejo o incluso imposible. Un sistema de enclavamientos moderno, robusto, que pueda garantizar una altísima disponibilidad, mantenibilidad y seguridad es indispensable para llevar a cabo estas tareas.

1.2. Principios de señalamiento ferroviario

El proceso de diseño del señalamiento requiere reglas claras y bien definidas sobre cuántas señales colocar, dónde colocar cada señal, bajo qué condiciones, de qué tipo deben ser las señales, cómo deben orientarse, etc. Lamentablemente, el criterio utilizado a la hora de definir el señalamiento dista de ser uniforme en los distintos países. La obligatoriedad de ciertas señales, la protección de ciertos elementos, la granularidad de la red o incluso el tener las reglas por escrito son factores que cambian al atravesar las fronteras de cada país. Esto implica, claramente, una barrera enorme al tratar de integrar las redes ferroviarias transnacionales, como en el caso Europeo durante formación de la Unión Europea [6, 15].

Para la realización de este trabajo se optó por recurrir al Instituto de Ingenieros en Señalamiento Ferroviario (IRSE, por sus siglas en inglés) [12] y a la Junta de Normas y Seguridad de la Industria Ferroviaria (RISSB, por sus siglas en inglés) [3, 5, 9]. Los reglamentos de diseño y definiciones de estas organizaciones son aceptadas por una gran cantidad de empresas del sector ferroviario y autoridades de gran peso. Entre ellas, por la agencia de transporte de Nueva Gales del Sur, en Australia (TfNSW, por sus siglas en inglés) [4]. De ésta última se recopilaron los siguientes principios de diseño ferroviario:

- (P₁) Principio de autoridad: la combinación de todas las rutas alcanza todo el tendido ferroviario.
- (P₂) Principio de claridad: las autoridades otorgadas o negadas no deben ser ambiguas.

- (P₃) Principio de anticipación: los conductores de trenes deben ser advertidos de los peligros cercanos con el suficiente tiempo para poder reaccionar.
- (P₄) Principio de granularidad: las rutas habilitan el uso de una pequeña porción de la infraestructura.
- (P₅) Principio de terminalidad: los conductores de trenes deben ser advertidos cuando se encuentren circulando próximos al final de la red.
- (P₆) Principio de infraestructura: los conductores de trenes deben ser advertidos de cualquier infraestructura dinámica o estática próxima.
- (P₇) Principio de no bloqueo: se debe evitar en todo momento que los trenes bloqueen el acceso a la infraestructura o ramificaciones a otros trenes, de ser posible.

La totalidad del análisis realizado en este trabajo se basa en los principios expuestos. Se puede deducir de los mismos que un sistema de señalamiento debe proteger cada elemento ferroviario (P₁, P₅, P₆), en cada dirección posible (P₃, P₇). Por lo tanto debemos considerar la posibilidad de que cada tramo de vía pueda ser transitado en ambos sentidos (P₂, P₄). Esto implica a su vez considerar todas las rutas posibles soportadas por la red ferroviaria, y no solamente las necesarias desde un punto de vista logístico.

1.3. Sistemas de enclavamiento

1.3.1. Tipos de sistemas de enclavamiento

Los primeros sistemas de enclavamiento implementados fueron mecánicos [16]. Estos utilizaban palancas, como las que se visualizan en la Figura 1.5, para comandar los cambios de vías y semáforos. Una palanca que habilite una ruta se encuentra bloqueada mecánicamente (enclavada) hasta que todas las palancas que afecten a los elementos ferroviarios correspondientes a esa ruta se encuentre en la posición que garanticen que la ruta es segura. De la misma manera, una vez habilitada una ruta, no es posible mecánicamente mover una palanca que habilite una ruta contraria y, por lo tanto, lleve a dos formaciones a colisionar frontal o lateralmente.



Figura 1.5: Sistema de enclavamiento mecánico basado en palancas.

Aunque hoy en día las tecnologías más modernas ya no utilizan bloqueos mecánicos, se sigue utilizando el término *enclavamiento*. En lugar de palancas y mecanismos de bloqueo, se utilizan circuitos lógicos, tanto eléctricos como electrónicos, y lógica programable que permitan garantizar el mismo objetivo: verificar condiciones previo a autorizar una ruta y prohibir la habilitación de rutas conflictivas con las actualmente activas.

A comienzos del siglo XX se empezaron a utilizar sistemas de enclavamiento electromecánicos en entornos urbanos y semi-urbanos [17]. Hoy en día este sistema es de los más utilizado en muchos países [16], sobretodo en áreas rurales donde los sistemas tardan más en actualizarse en comparación con las áreas urbanas. Un sistema de enclavamiento electromecánico se basa en relés (Figura 1.6) y circuitos de vía. Los circuitos eléctricos implementados con relés determinan la lógica del enclavamiento y son necesarios unas pocas decenas de relés para operar una derivación ferroviaria, pero cientos a miles para una estación terminal de gran tamaño [18].



Figura 1.6: Sistema de enclavamiento electromecánico basado en relés.

Los sistemas de enclavamiento electromecánicos son comandados por un operario mediante un panel de control (Figura 1.7). El operario solicita al sistema de enclavamiento las rutas que el conductor ferroviario necesita para transitar por la vía. El sistema de enclavamiento solamente habilitará las rutas seguras, indicando al conductor que puede avanzar mediante un semáforo de aspecto verde o amarillo. En caso de que la ruta no sea habilitada, el sistema *enclavará* los estados de los elementos ferroviarios pertenecientes a esa ruta y le indicará al conductor que no puede continuar su recorrido con un semáforo de aspecto rojo. El operario puede volver a pedir la ruta si las condiciones del sistema cambiaron.



Figura 1.7: Panel de control central de un sistema de enclavamientos.

En la actualidad, los sistemas de enclavamiento más modernos son electrónicos, basados en microprocesadores, PLCs (Controlador Lógico Programable, del inglés Programmable Logic Controllers), o FPGAs (Matriz de puerta programable en campo, del inglés Field-programmable Gate Array) [19-32]. Estas tecnologías presentan la ventaja de ser reprogramables una vez instaladas, lo que aporta flexibilidad al diseño, escalabilidad y robustez [33-44]. Además, ocupan mucho menos espacio que su contraparte electromecánica, reduciendo los costes de infraestructura, consumo eléctrico y mantenimiento. Se utilizan principalmente en redes de procesamiento distribuido, debido a su integración de forma nativa con protocolos de comunicación industrial diseñados para entornos ferroviarios [45-47]. Adicionalmente, los sistemas electrónicos pueden ser fácilmente auditados y redundados, para aumentar su fiabilidad y comprobar su nivel de seguridad [48-55].

1.3.2. Tablas de enclavamientos

Históricamente, tanto los enclavamientos mecánicos como los electromecánicos se han definido mediante tablas de enclavamientos [10-12, 44, 56], las cuales luego 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.

Cada ruta es definida junto con los elementos ferroviarios que la condicionan y los estados que deben tener los mismos para que la ruta sea habilitada. Además, se explicita cada una de las rutas que entrarían en conflicto con la ruta en cuestión. A modo de ejemplo se presenta en la Figura 1.8 como se realiza la asignación de rutas en un cambio de vías.

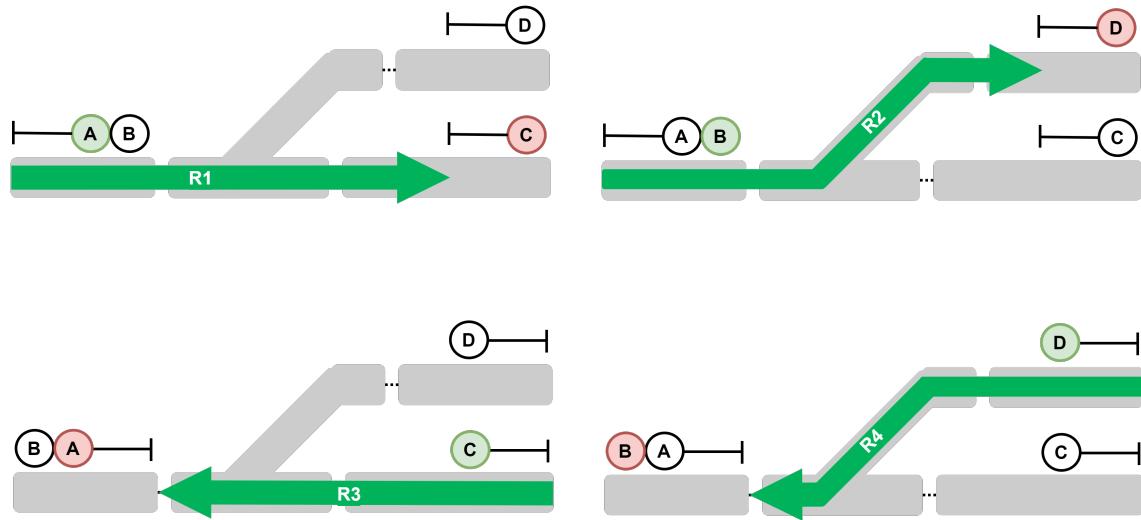


Figura 1.8: Ejemplo de asignación de rutas de izquierda a derecha (arriba) y de derecha a izquierda (abajo).

Si el trazado de vías fuese utilizado para circular únicamente de izquierda a derecha, podemos definir únicamente dos rutas. La primera será la ruta R1, desde la señal A hasta la señal C. La segunda ruta R2 se define desde la señal B hasta la señal D, utilizando el cambio de vía a reverso. Ambas rutas quedan definidas en la Tabla 1.1.

Tabla 1.1: Tabla de enclavamientos (rutas de izquierda a derecha).

Ruta	Señal de entrada	Señal de salida
R ₁	Señal _A	Señal _C
R ₂	Señal _B	Señal _D

De forma análoga, podríamos analizar el mismo trazado de vías asumiendo que las formaciones circularán estrictamente de derecha a izquierda, definiendo otro nuevo conjunto de rutas. Sumando así las rutas R3 y R4 como se visualiza en la Tabla 1.2

Tabla 1.2: Tabla de enclavamientos (rutas de derecha a izquierda).

Ruta	Señal de entrada	Señal de salida
R ₃	Señal _C	Señal _A
R ₄	Señal _D	Señal _B

Ambas tablas de enclavamiento (Tabla 1.1 y Tabla 1.2) son válidas, dependiendo del uso que se le quiera dar a la infraestructura. Incluso podemos notar que ambas tablas están incompletas porque una tabla no contempla los casos que contempla la otra y viceversa. Por lo tanto, es posible

definir una nueva tabla de enclavamientos como la conjunción de ambas, como se muestra en la Tabla 1.3, especificando las rutas conflictivas.

Tabla 1.3: Tabla de enclavamientos completa.

Ruta	Señal de entrada	Señal de salida	Rutas conflictivas
R_1	Señal _A	Señal _C	R_2, R_3 y R_4
R_2	Señal _B	Señal _D	R_1, R_3 y R_4
R_3	Señal _C	Señal _A	R_1, R_2 y R_4
R_4	Señal _D	Señal _B	R_1, R_2 y R_3

1.3.3. Enfoque funcional

A la hora de abordar el análisis de las redes ferroviarias, hemos encontrado que, a grandes rasgos, existen dos estrategias muy diferenciadas: el enfoque funcional y el enfoque geográfico [44, 57], cada una con sus fortalezas y debilidades. Ambos enfoques se asemejan a la discusión de arquitecturas CISC (del inglés, Complex Instruction Set Computing) vs RISC (del inglés, Reduced Instruction Set Computing). El enfoque funcional centraliza las decisiones en un único módulo complejo, mientras que el enfoque geográfico distribuye las decisiones en pequeños módulos de funciones reducidas.

En el enfoque funcional las decisiones se basan en la 'tabla de enclavamientos', que define cada ruta que puede ser solicitada por el operario. Encontramos entonces, el primer gran problema del enfoque funcional: como se exemplificó en la Sección 1.3.2, las soluciones dadas por una tabla de enclavamientos no son únicas, dependen del itinerario que se quiera establecer sobre la base de la infraestructura. Ese itinerario puede variar con el tiempo, haciendo necesario añadir o eliminar algunas rutas, lo que obliga a tener que volver a implementar y certificar todo el sistema. Además, muchas de las soluciones no contemplan todas las rutas posibles, por lo que el diseño es incompleto y siempre existirá el riesgo de tener que repetir todo el proceso desde cero [44].

El segundo problema radica en que, al ser el enfoque funcional una arquitectura que prioriza la funcionalidad a nivel macro, abstrayéndose de la topología de la red, la concurrencia de las rutas no está garantizada. Es decir, si N rutas dependen del estado de un elemento en común, no puede garantizarse ni que el resultado de cada ruta se calcule en paralelo, ni tampoco que el resultado de cada ruta se obtenga en simultáneo. Para solucionar esto, es necesario repetir N veces el elemento en cuestión, asociando uno a cada ruta que condiciona, incrementando la complejidad del diseño [44].

Claramente, ya que el enfoque funcional no garantiza la concurrencia del sistema, es necesario tomar medidas que terminan aumentando la cantidad de recursos necesarios. A medida que la complejidad de la red ferroviaria aumenta, incrementa a su vez la interrelación de sus elementos y la necesidad de mantener la concurrencia del sistema termina generando que la memoria utilizada crezca exponencialmente [44].

Por lo tanto, el enfoque funcional es de muy fácil implementación para topologías pequeñas y tradicionalmente se lo considera como el único enfoque por defecto. Sin embargo, presenta grandes falencias a la hora de resolver sistemas de mediana y alta complejidad. El enfoque funcional no posee concurrencia de forma directa, desperdicia mucha memoria y su solución muchas veces resulta incompleta. Un sistema de enclavamientos diseñado con un enfoque funcional difícilmente será mantenible en el tiempo ni mucho menos escalable o fácil de actualizar.

1.3.4. Enfoque geográfico

En el enfoque geográfico, el énfasis está puesto en la interacción entre los componentes a partir de su posición en la red y no en su funcionalidad a nivel de sistema [58-61]. Por consiguiente, el enfoque geográfico 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 mayor esfuerzo de desarrollo. No obstante, es mucho más escalable a topologías de cualquier tamaño al hacer un uso más eficiente de los recursos al tener concurrencia directa [2, 62, 63]. Además, es posible añadir nuevos componentes en el futuro, haciéndolo mucho mas flexible de ser aplicado en otros países con diferentes elementos ferroviarios. Este enfoque independiza la estrategia de diseño de la ubicación, haciendo posible replicar de forma sistemática el mismo conjunto de herramientas en cualquier topología [64-66].

Aunque el concepto de ruta sigue existiendo, ya no es el foco central del proceso de diseño. La tabla de enclavamiento deja de ser una entrada del proceso y pasa a ser un subproducto [66]. La herramienta que realice el análisis de la red ferroviaria deberá obtener todas las rutas posibles que soporta esa topología y registrarlas en una tabla de enclavamientos. La tabla de enclavamientos del enfoque funcional debe estar contenida en la tabla de enclavamientos del enfoque geográfico; ambos enfoques deben ser consistentes.

1.4. Estándares de seguridad

1.4.1. Estándares EN-50126, EN-50128 y EN-50129

El estándar internacional IEC 61508 [67-73] es la base para definir un conjunto de estrategias y buenas prácticas para ser aplicadas en el diseño de sistemas críticos, eléctricos y electrónicos, en diversas industrias [27, 28, 35, 49, 74-78]. Entre las industrias alcanzadas por los lineamientos básicos establecidos en la IEC 61508 se encuentran la industria nuclear, la industria automotriz y la industria ferroviaria, como se puede visualizar en la Figura 1.9.

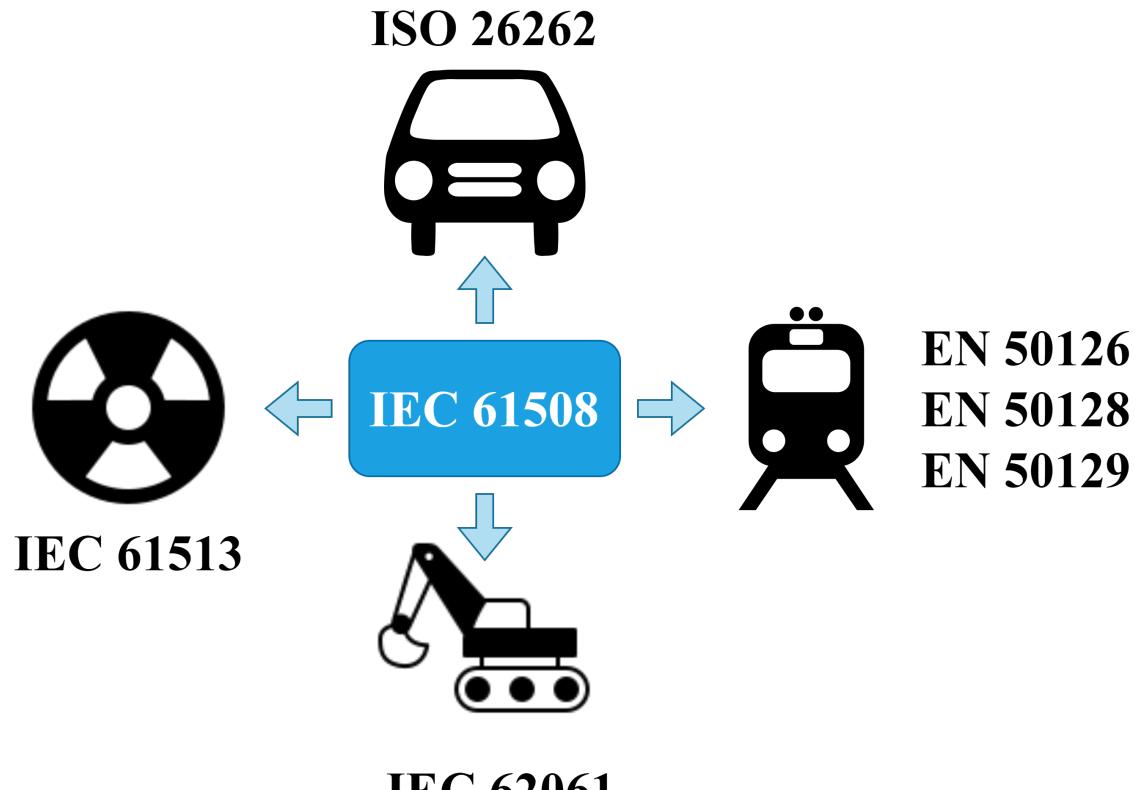


Figura 1.9: Estándar IEC 61508.

Dentro del universo de estándares ferroviarias, nos enfocaremos principalmente en tres de ellas: EN-50126 [79-83], EN-50128 [83, 84] y EN-50129 [82, 85]. Una falla en un sistema crítico puede poner en peligro cientos de vidas humanas y/o dañar la costosa infraestructura del sistema. Es por eso que los sistemas de enclavamiento deben cumplir los estrictos parámetros de fiabilidad, disponibilidad, mantenibilidad y seguridad (RAMS, del inglés Reliability, Availability, Mantenability and Safety) [86], durante todo el ciclo de vida definido en el estándar EN-50126 [79, 87, 88].

Adicionalmente, el estándar EN-50126 define el nivel de integridad de seguridad (SIL, del inglés Safety Integrity Level). Este parámetro no es del sistema en su conjunto, sino que se asocia a las funciones de seguridad asociadas al sistema, pudiendo tener diferentes SILs dentro de un mismo dispositivo. Se definen cuatro niveles SILs, siendo el cuatro el nivel mas alto y, por lo tanto, el que tiene una tasa de fallas menor. Aunque existen diversos factores que impactan en la determinación del SIL, podemos tomar como referencia el criterio definido en la Tabla 1.4 en función de la probabilidad de fallas por hora (PFH) del sistema funcionando de forma continua.

Tabla 1.4: SIL en función de la Probabilidad de Fallas/Hora (PFH).

SIL			
4	10^{-8}		10^{-9}
3	10^{-7}		10^{-8}
2	10^{-6}	$>$	10^{-7}
1	10^{-5}		10^{-6}

El estándar EN-50128 establece las buenas prácticas, metodologías y técnicas aplicables en el área de software para lograr el SIL objetivo. Algunas de estas técnicas pueden ser obligatorias, altamente recomendadas, recomendadas, desaconsejadas o prohibidas [84, 89-95]. En algunos casos es obligatorio elegir entre diferentes combinaciones de metodologías optativas. Podemos encontrar entre estas: evitar el uso de punteros, elegir un lenguaje fuertemente tipado, modularizar el código, prohibición de usar memoria dinámica, entre otras [84, 96-98]. Este estándar solo se aplica si el sistema incluye un microprocesador con software embebido.

El estándar EN-50129 es el equivalente en hardware al estándar EN-50128. Las metodologías definidas en EN-50129 se centran en la elección de componentes electrónicos [99-104], compatibilidad electromagnética y dos conceptos fundamentales: redundancia [48, 49, 51-53, 105, 106] y diversidad [41, 45, 54, 55, 107, 108].

1.4.2. Redundancia y diversidad en sistemas ferroviarios

La redundancia en sistemas críticos se define mediante la abreviatura NooM (N de M, del inglés, N out of M). Donde M representa la cantidad de módulos de medición/decisión que posee el sistema y N la cantidad de módulos que deben funcionar correctamente para que el sistema opere normalmente [35, 47, 68, 70, 72, 88, 109]. En la Figura 1.10 se puede apreciar un sistema con redundancia 2oo3, en el cual se tendrá una salida correcta siempre que se tenga a lo sumo un fallo por vez, no acumulativo. El sistema determina, mediante votación, la respuesta correcta a presentar como salida.



Figura 1.10: Sistema con redundancia 2oo3 y diversidad.

La tasa de fallas de los sistemas utilizados debe ser lo más pequeña posible. Existen fallas de causa común asociadas al fabricante de componentes, defectos eléctricos en los materiales, desperfectos en el software replicados en cada uno de los módulos, etc [35, 37, 38, 41, 45, 54, 55, 67, 73, 75, 88, 103, 105, 106, 110-113]. Para mitigar las fallas de causa común y robustecer el sistema, se aplica el concepto de diversidad [41, 45, 54, 55, 107, 108]. En el caso de la Figura 1.10, esta diversidad está representada en los módulos de diferentes colores: azul, rojo y verde. De esta forma se busca simbolizar que los tres sistemas cumplen la misma función, pero de manera diferente (distintas plataformas y/o distinto lenguaje de programación o incluso diferente equipo de desarrollo). Los tres módulos tendrán, por lo tanto, tasas de falla distintas. Se asume, además, que los módulos pueden tener fallas simultáneas pero de probabilidad mucho más baja que la tasa de fallas inherente a cada módulo.

De la investigación realizada surge que el 66 % de las empresas utiliza una redundancia 2oo2 o 2oo3 para alcanzar los niveles de seguridad requeridos [114-127]. Solo una pequeña porción de las mismas utiliza redundancias 1oo2 [115] o 2oo4 [116]. En consecuencia, se puede afirmar que una redundancia 2oo2 o 2oo3 es representativa de los sistemas analizados y puede utilizarse como esquema de partida para un diseño propio.

1.5. RailTopoModel y RailML

1.5.1. RailTopoModel

En 2016 la UIC (del inglés, International Union of Railways) publicó el Standard 30100 [128] en el cual definió un formato de intercambio de datos ferroviarios llamado RailTopoModel [63, 66, 129-131]. RailTopoModel es un modelo topológico de infraestructura ferroviaria basado en grafos. El modelo abarca tres tipos de niveles, la topología de la red, objetos materiales, objetos inmateriales y objetos lógicos [128], tal como se ilustra en la Figura 1.11 (adaptada al español de [128]).



Figura 1.11: Alcance del modelo de UIC RailTopoModel.

Cada uno de estos elementos posee propiedades y características propias, que pueden ser físicas o abstractas. Por ejemplo, los objetos materiales pueden ser adimensionales (señales, balizas, etc), unidimensionales (vías, plataformas, etc) o bidimensionales (estaciones, túneles, etc). Debido a cómo fue diseñado, RailTopoModel es un modelo muy apropiado para implementar un enfoque geográfico [62-65, 130, 132].

1.5.1.1. Paquete base

El estándar RailTopoModel se divide en cuatro paquetes: la base, la topología, el posicionamiento y la red de entidades. La base incluye toda la información de alto nivel de la infraestructura que son comunes a toda la red [63]. Por ejemplo, el sistema de alimentación eléctrica utilizado puede ser por tercer riel o catenarias y este no cambia a lo largo de toda la red. La topología y la red de entidades describen las secciones ferroviarias y su interconexión, formando la estructura principal de la red sobre la cuál se define la infraestructura. El posicionamiento establece las diferentes formas de definir la ubicación de los elementos ferroviarios, tanto relativas a las secciones como absolutas a un sitio específico.

1.5.1.2. Modelo de grafos

En el pasado, otros estudios han tratado de modelar el trazado ferroviario utilizando teoría de grafos [57, 59-61]. Pero todos ellos definían a las vías como las aristas del grafo y los cambios de vías como los nodos, lo cual dificultaba enormemente la inclusión de otros elementos ferroviarios como las plataformas, los pasos a nivel o los semáforos. RailTopoModel se apega mas a la teoría clásica de grafos y define como nodos a la unidad mínima de recursos físicos, llamándolos netElements y a la conexión física entre ellos como aristas, llamándolos netRelations [128].

Un netElement debe contener un tramo de vía en su totalidad o varios tramos, pero un tramo de vía no puede tener varios netElements asociados. Opcionalmente, un netElement puede tener además cualquier otro elemento ferroviario asociado como plataformas, semáforos o balizas. Para entender mejor como se materializa este concepto se tiene la Figura 1.12 (adaptada al español de [128]) que ejemplifica el modelado en grafos de un cambio de vías simple.

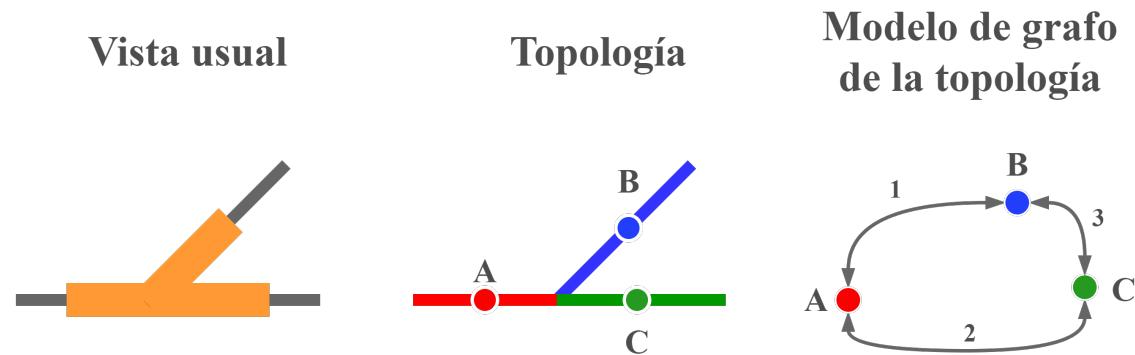


Figura 1.12: Modelado en grafos de un cambio de vías simple.

En la Figura 1.12 se tiene un cambio de vías simple que conecta una vía de circulación (horizontal) con una vía de maniobras (oblicua). El netElement asociado a la vía principal antes de la bifurcación lo llamaremos A. Este netElement es también al que se asocia al objeto del cambio de vías simple. El netElement asociado a la vía de maniobras lo denominamos B y al asociado a la vía de continuación de la circulación lo denominamos C.

En la representación del modelo de grafos, el netRelation 1 relaciona el netElement A y B de forma biyectoria. Es decir, A está relacionado con B y B está relacionado con A. De la misma forma se asignan los netRelations 2 y 3. Sin embargo, no hay que confundir que dos vías estén conectadas con que un tren puede circular por ellas. Una formación podría circular por el tramo A-C, A-B, B-A y C-A, pero no podrá circular desde B hacia C sin pasar por A o viceversa. Es físicamente imposible que un tren realice ese movimiento de una sola maniobra. Para circular desde B hacia C primero deberá finalizar el tramo B-A, modificar la posición del cambio de vías y recorrer el tramo A-C. A la propiedad asignada a los netRelation que son físicamente transitables se la denomina navegabilidad y es una característica esencial de las redes de grafos que modelan redes ferroviarias. En este ejemplo, solo los netRelation 1 y 2 poseen navegabilidad. Los cambios de vías, sean estos simples, dobles o en tijeras, son los únicos elementos ferroviarios que alteran de forma dinámica la red ferroviaria y, por lo tanto, los únicos que afectan al parámetro de navegabilidad en los netElements adyacentes.

1.5.1.3. Topología y niveles

La topología del modelo es una parte fundamental del estándar. RailTopoModel incluye el "principio de agregación", por el cual los elementos pueden ser agrupados en entidades mas gran-

des. En la Figura 1.13 se puede visualizar la estructura de capas propuesta por RailTopoModel (adaptada al español de [128]).

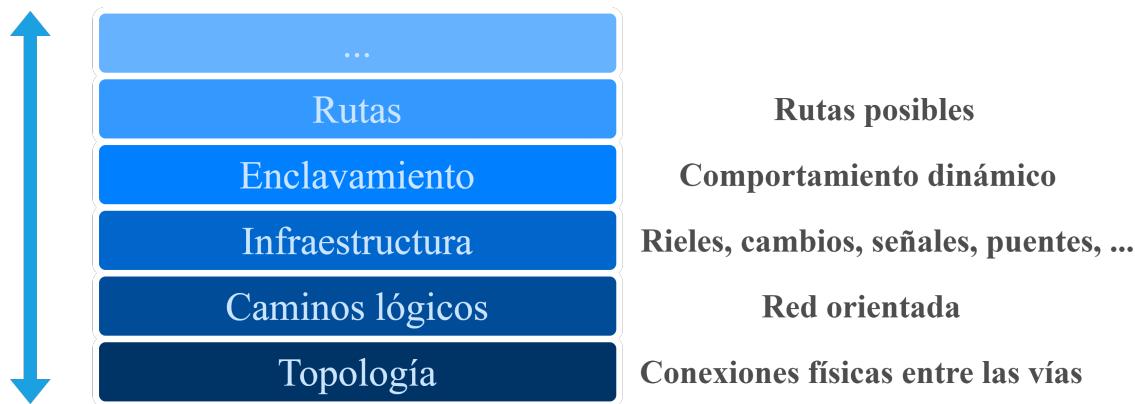


Figura 1.13: Estructura de capas de RailTopoModel.

La topología de la red está compuesta por los nodos (netElements) y las aristas (netRelations) que los conectan entre sí, lo cual constituye el nivel microscópico de la red, de acuerdo con la terminología de RailTopoModel. Cada nodo representa un tramo de vías que puede tener ciertos elementos ferroviarios asociados o ninguno. A su vez, los nodos pueden ser agrupados en diversos caminos lógicos, que son el conjunto de nodos cuyas relaciones y navegabilidad les permite constituir un camino físico entre ellos [131].

A medida que se agrupan más y más cambios de vías junto con las plataformas y máquinas de cambios se constituye un punto de operación. La descripción basada en puntos de operación es a nivel "mesoscópico", como se muestra en la Figura 1.14 (adaptada al español de [128]), y es utilizado en logística. Las secciones de vía que no incluyen plataformas en las cuales las formaciones puedan detenerse se denominan secciones de líneas, o simplemente "líneas" dentro del modelo de RailTopoModel. La descripción que incluye tanto los puntos de operación como las secciones de líneas es a nivel "macroscópico" [130]. Esta simplificación de la red es de gran importancia, ya que es ampliamente utilizada en los mapas ferroviarios: los puntos de operación son las estaciones y las secciones de línea son las vías que las comunican.

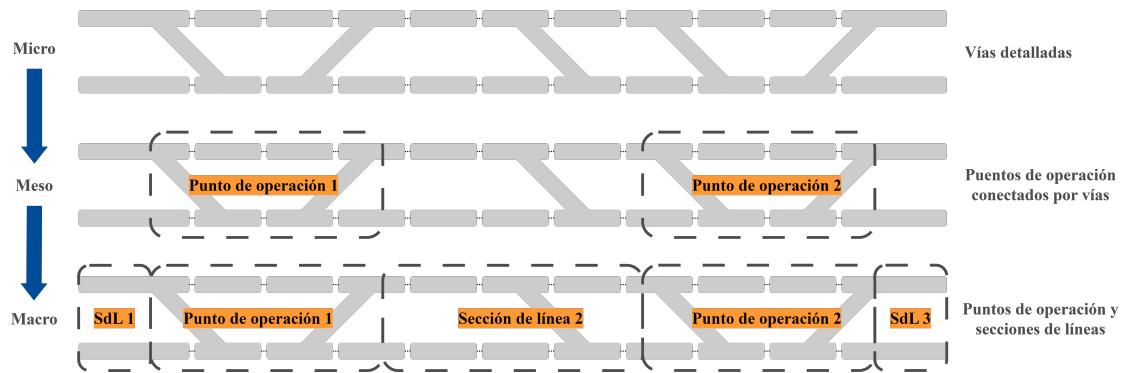


Figura 1.14: Niveles microscópico, mesoscópico y macroscópico.

Las instalaciones y sus propiedades constituyen todos los elementos ferroviarios asociados a un nodo. Estos representan elementos físicos del mundo real, que pueden ser estáticos o dinámicos.

Los elementos estáticos como los puentes, curvas y estaciones no alteran sus propiedades en ningún momento. Los elementos dinámicos como los pasos a nivel, cambios de vías o señales tienen algunas propiedades fijas, como la posición física del elemento, pero otras variables, como la posición mecánica de alguna de sus piezas o el estado eléctrico de sus circuitos [63, 131].

Un sistema de enclavamiento relaciona todos los módulos previamente mencionados. El sistema de enclavamientos modificará el estado de los elementos dinámicos, basados en el estado actual de los mismos, sometido a las restricciones impuestas por los elementos estáticos, buscando habilitar los caminos lógicos mas cortos y seguros entre dos puntos A y B.

Finalmente, las rutas permitidas se obtienen a partir del estado de los elementos dinámicos decidido por el sistema y de definir el camino óptimo entre A y B que no comprometa la infraestructura del sistema. Todas las restricciones impuestas por las capas inferiores (caminos lógicos posibles, limitaciones de la infraestructura o estados previos que sean incompatibles con lo pedido) terminan emergiendo como un conjunto de rutas posibles de ser utilizadas, en detrimento de otras que, en ese instante de tiempo, no podrán ser habilitadas hasta que el estado del sistema se modifique.

Como se puede apreciar, en este modelo, las rutas son una consecuencia de la infraestructura que se tiene y de los estados anteriores del sistema, producto de las rutas previamente pedidas. Un análisis completo de la topología e infraestructura permitiría obtener todo el conjunto de rutas posibles, para cualquier estado alcanzable por el sistema [131].

1.5.1.4. Posicionamiento

RailTopoModel utiliza diferentes sistemas de posicionamiento: intrínseco, geográfico y esquemático [131, 133]. Las coordenadas intrínsecas se encuentran en el rango 0 a 1, relativas a la posición dentro del netElement. Estas coordenadas son obligatorias, junto con el largo asociado al netElement, independientemente de si se definieron o no las demás coordenadas. Las coordenadas geográficas son coordenadas absolutas, por ejemplo de un sistemas GPS. Finalmente las coordenadas esquemáticas son relativas a la posición del elemento dentro del archivo que modele ese sistema, muchas veces utilizadas en herramientas de software para posicionar los elementos en una interfaz gráfica. Se explicarán ejemplos de como analizar esta clase en la Sección 2.1.3.

1.5.2. RailML 3.0

railML [131-135] (del inglés, Railway Markup Language) es un estándar abierto de datos basado en XML, para intercomunicar aplicaciones ferroviarias. El estándar railML fue desarrollado por las principales empresas de la industria ferroviaria a partir del año 2001 [135]. Las primeras dos versiones del estándar se diseñaron en base al enfoque funcional, pero en 2017 se lanzó railML 3.0 [63, 131, 134], que adopta gran parte de los conceptos de RailTopoModel y los expande, al incorporar nuevos elementos que consideran las necesidades de las empresas ferroviarias que lo adoptan. Debido a la incorporación de RailTopoModel, railML 3.0 adopta un enfoque puramente geográfico [131].

El estándar railML 3.0 posee cinco módulos principales:

- Common (CO) [136]: similar al módulo base de RailTopoModel. Contiene datos transversales a todo el sistema, como el autor del archivo, el sistema eléctrico de alimentación, la versión de railML utilizada, etc.
- Infrastructure (IS) [137]: incorpora al módulo de topología de RailTopoModel, con sus netElement y netRelations. Ademas, incorpora el sistema de coordenadas, la geometría y todos los elementos ferroviarios. En estos últimos solo admite datos estáticos como la posición, tamaño y demás atributos invariantes en el tiempo.

- Interlocking (IL) [138]: incorpora los atributos dinámicos de los elementos ferroviarios mencionados en la infraestructura y agrega el listado de rutas.
- Rollingstock (RS) [139]: incorpora toda la información relativa al material rodante: coches, vagones, locomotoras, formaciones combinadas, etc.
- Timetable and Rostering (TT) [140]: incorpora detalles logísticos de la red ferroviaria.

A lo largo de esta tesis doctoral nos enfocaremos exclusivamente en los tres primeros módulos. Los cuales serán analizados detalladamente a medida que sea necesario introducirlos.

1.5.3. Uso del estándar railML en la industria ferroviaria

El estándar railML es promovido por empresas de gran peso en la industria ferroviaria como Siemens, Thales, Alstom, CAF, ADIF y Toshiba, que concentran la mayoría de la cuota de mercado global [141]. Adicionalmente, diversas instituciones y organismos ferroviarios a nivel estatal y nacional hacen uso del estándar railML en sus desarrollos ferroviarios [141], tales como: Queensland Rail, Transdev Deutschland, Transperth, Saudi Railway Company y Transport for New South Wales, entre otras.

En sus primeros años de vida railML experimentó varios cambios, pero no fue hasta su versión 3.0 con enfoque geográfico que el uso del estándar creció exponencialmente. Entre el 40 y el 60% de sus usuarios adoptaron el estándar en los últimos siete años [141].

Podemos encontrar las herramientas más diversas basadas en railML: analizadores de infraestructura [142], planificador logístico para material rodante [143], visualizadores de datos [144], planificadores de infraestructura [145] y visualizadores/simuladores de infraestructura enclavamiento [146]. Muchas de ellas certificadas e intercompatibles entre sí. Aunque la mayoría son herramientas de código cerrado, el estándar railML es abierto y sigue un principio bottom-top: todas las necesidades de la industria son tenidas en cuenta para ser incorporadas en nuevas versiones del estándar, siguiendo el exitoso modelo del estándar USB, Bluetooth y GPRS [147-149].

1.6. Estado del arte

1.6.1. Empresas del sector ferroviario

Al requerir ingenieros y técnicos más especializados, el conocimiento requerido tanto para diseñar como para mantener y operar los sistemas de enclavamientos electrónicos se concentra en menos actores. La tendencia mundial en las últimas décadas ha sido el depender de soluciones cerradas provistas por alrededor de una docena de empresas, muchas veces incompatibles entre sí. Mas del 80% del material rodante que circula en el mundo proviene de alguna de estas doce empresas [150]:

- CRRC Railway (China) [121]
- Alstom (Francia) [115]
- Bombardier Transportation (Canada) [118]
- CAF (España) [122]
- Hitachi (Japón) [116]
- Transmashholding (Rusia) [123]
- Hyundai Rotem (Corea del sur) [124]

- Siemens Mobility (Alemania) [114]
- Thales (Reino unido) [117]
- General Electric Transportation (EEUU) [125]
- Caterpillar (EEUU) [126]
- Stadler Rail (Suiza) [127]

Al lado de cada empresa se puede ver su país de origen. Cuatro de esos países (China, EE.UU., Canadá y Rusia) son las naciones con mayor extensión territorial, para lo cual un sistema ferroviario robusto es esencial.

1.6.2. Antecedentes de trabajos académicos en la temática

Dada la variedad de topologías (algunas mencionadas en la Sección 1.1), es deseable automatizar tanto el análisis de la red como el proceso de diseño del señalamiento. De esta manera, se puede reducir el error humano que puede afectar el proceso en cualquiera de sus etapas: análisis, diseño, implementación, verificación y validación.

Varios artículos abordan la automatización de las tablas de enclavamiento [65, 151-154], que son las tablas que indican cuáles elementos ferroviarios utilizan cada ruta que la formación puede solicitar al sistema. También podemos encontrar trabajos respecto de la automatización de otros elementos como el comportamiento de los enclavamientos [62, 90, 155-159], las rutas [160], el señalamiento de los pasos a nivel [161], topologías simples [162, 163], así como sus simulaciones [164, 165].

Existen, además, diversos estudios acerca de la verificación y validación de los sistemas ferroviarios utilizando métodos formales [108, 163, 166-188]. Algunos emplean herramientas que analizan y validan el modelo tales como NuSMV [15, 163, 189, 190], SafeCap [188, 191, 192], o soluciones completas como el B-method [193-196]. No obstante, luego de recopilar mas de 150 artículos [197], no hemos encontrado una herramienta (o conjunto de herramientas) que realice el análisis de una red ferroviaria arbitraria, generando automáticamente su señalamiento completo y la implementación automática de su sistema de enclavamiento asociado. Los pocos casos que aventuran análisis automáticos lo realizan solo para topologías simples de un único elemento [198, 199], mientras que en este trabajo se busca la generalización para cualquier red ferroviaria con cualquier combinación de diversos elementos ferroviarios.

1.6.3. GICSAFe y los trabajos realizados

En 2015 se creó el CONICET-GICSAFe [200], cuyas siglas corresponden al Grupo de Investigación en Calidad y Seguridad de las Aplicaciones Ferroviarias, conformado por docentes e investigadores de una decena de universidades e instituciones públicas argentinas. El grupo desarrolló 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 transfirió en su totalidad a Trenes Argentinos [201], que es la Sociedad del Estado que opera las líneas Roca, Sarmiento, Mitre, San Martín y Belgrano Sur, entre otras. En la Figura 1.15 se ilustran todos los logros e hitos alcanzados por GICSAFe en relación con el desarrollo de un sistema de enclavamientos desde 2018 y los objetivos a futuro hasta 2025.

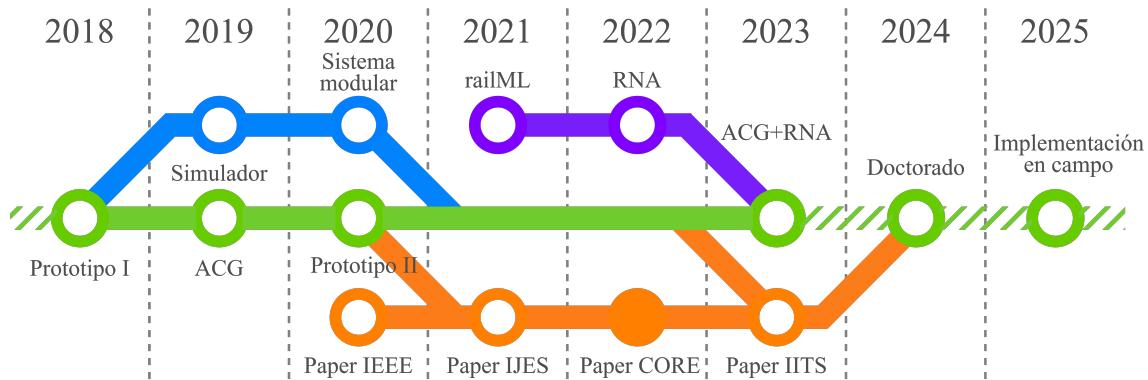


Figura 1.15: Trabajos realizados e hitos del proyecto.

En el año 2018 GICSAFe desarrolló un monitor de barreras ferroviarias instalado en la línea Roca, proyecto que ganó el premio Innnowar 2018 [202, 203] y permitió ganar experiencia en el área ferroviaria. Desde 2018 se tuvieron varias reuniones con diferentes funcionarios y profesionales de Trenes Argentinos. En particular, 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.

En el marco de la Maestría de sistemas embebidos, culminada en 2020, se desarrolló un primer prototipo de sistema de enclavamiento con código generado automáticamente en base a un modelo de grafos. Los resultados fueron publicados en IEEE Latin American Transaction en 2020 [44] y en International Journal of Embedded Systems en 2022 [204]. En simultáneo, el ingeniero Lucas Dórdolo, miembro de GICSAFe, presentó una actualización del sistema monitor de barreras: el sistema modular [205], que permitió la lectura y transmisión de datos de la red ferroviaria en tiempo real. Adicionalmente, miembros de GICSAFe pertenecientes a UTN-Haedo comenzaron el desarrollo de un simulador de enclavamientos en tiempo real, que permitió resolver varias cuestiones técnicas de la implementación final del sistema de enclavamiento [56].

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 testeada en el ámbito de los sistemas críticos. Además del intercambio de conocimientos y la puesta en común de estrategias a utilizar, se aprovechó todo lo posible la amplia experiencia que ellos brindaron a GICSAFe.

A mediados de 2020, ya formalmente inscripto en el doctorado, se comenzó el desarrollo de una biblioteca compatible con railML 3.0, lo cual se incorporó al primer analizador de redes ferroviarias en 2022, cuyos resultados se publicaron en IEEE Transactions on Intelligent Transportation Systems en 2023 [206]. La verificación y validación automática del sistema mediante métodos formales es parte del trabajo de doctorado del ingeniero Santiago Germino [207], iniciado en 2021. La integración de todas estas herramientas se aplicará en un sistema de enclavamientos a ser instalado en una playa de maniobras real en 2025.

Este proyecto no podría llevarse a cabo con las herramientas disponibles actualmente, mencionadas en la Sección 1.6.2. Es necesario ampliar el estado del arte, aplicando nuevos conceptos y desarrollando nuevas herramientas que permitan automatizar el diseño e implementación del sistema de enclavamientos, para cualquier locación, según la descripción de un lenguaje ferroviario estándar.

1.7. Herramienta propuesta

El objetivo de esta tesis es el desarrollo de una herramienta que realice automáticamente tanto el diseño del señalamiento como la implementación del sistema de enclavamiento en una plataforma electrónica. El flujo de trabajo mostrado en la Figura 1.16 introduce el Analizador de Redes Ferroviarias (RNA, del inglés, Railway Network Analyzer) y el Generador Automático de Código (ACG, del inglés, Automatic Code Generator). Cada flecha indica las relaciones entre los diferentes bloques. El RNA se enfoca principalmente en el comportamiento estático del sistema, mientras que el ACG se enfoca en el comportamiento dinámico. Ambos son explicados a profundidad en el Capítulo 2 y Capítulo 3, respectivamente.

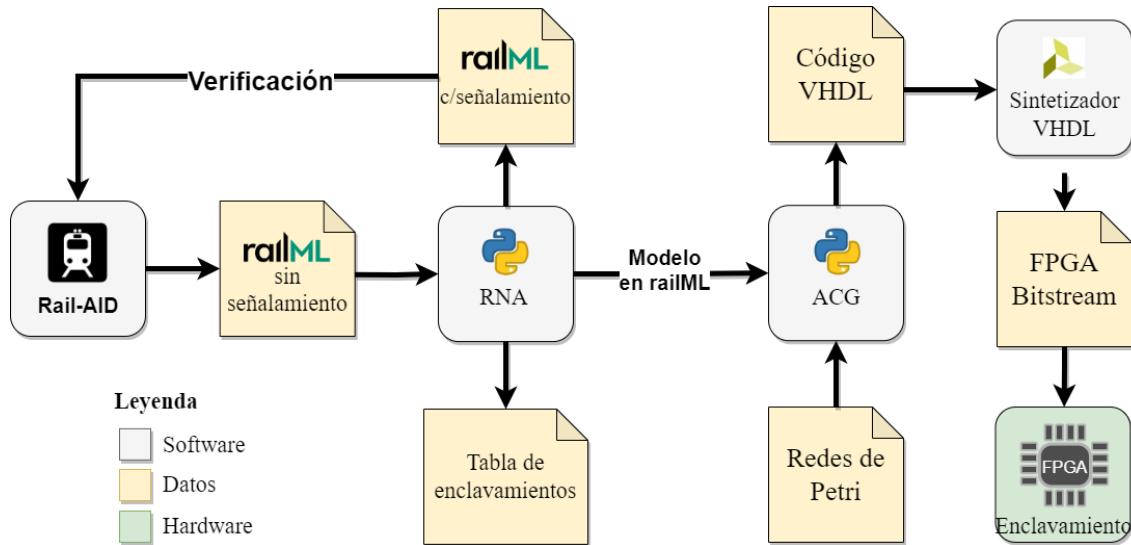


Figura 1.16: Flujo de trabajo de la presente tesis.

El RNA importa un archivo en formato railML que describe el sistema, con o sin señalamiento previo. Luego el RNA analiza la topología de la red, detecta todos los elementos ferroviarios involucrados y genera el señalamiento necesario para que la red sea segura. Finalmente, el RNA exporta un nuevo archivo en formato railML con el nuevo señalamiento, además de la tabla de enclavamientos del sistema, con todas las rutas soportadas por esa red ferroviaria. El nuevo archivo railML es utilizado por el ACG para generar código en VHDL [204] (del inglés, Very High-Speed Integrated Circuit Hardware Description Language) automáticamente para ser sintetizado en una plataforma FPGA [20, 25, 28, 34, 51] (del inglés, Field Programmable Gate Arrays).

El proceso de validación incluye la validación de la sintaxis del archivo railML generado por el RNA, la validación de la tabla de enclavamiento generada automáticamente por el RNA y, finalmente, la validación del señalamiento generado conforme a los principios de señalamiento (Sección 1.2). Para implementar automáticamente un sistema de enclavamiento, el ACG toma este diseño de señalamiento estático generado automáticamente y un modelo de comportamiento dinámico en redes de Petri.

1.7.1. Enfoque aplicado

Durante la Maestría en Sistemas Embebidos se desarrolló un primer prototipo del ACG en base a un rudimentario modelo de redes de grafos [204]. Fue este modelo de grafos el que permitió probar el ACG con una amplia variedad de topologías, debido a la flexibilidad, linealidad y escalabilidad

del modelo [128, 130, 131]. En esta etapa temprana del proyecto ya se había adoptado un enfoque geográfico.

La única desventaja de la elección temprana de un enfoque geográfico fue la, aparente, inexistencia de herramientas o soportes formales, para lo cual era necesario desarrollar todo desde cero sin un estándar que lo sostenga. Aún así, una vez desarrollada la herramienta se podía reutilizar innumerables veces, por lo que las ventajas a largo plazo eran mayores que realizar un diseño a mano, a medida de cada locación. Al descubrir la existencia de railML, rápidamente se comenzó la migración del ACG para ser compatible con el estándar.

El desarrollo del RNA fue posterior al desarrollo del ACG, y fue completamente en línea con el estándar railML desde un inicio. Al ser compatibles con railML, tanto el RNA como el ACG deben ser compatibles también con el modelo planteado por RailTopomodel. Debido a esto, el desarrollo de la herramienta siguió indefectiblemente los lineamientos de un enfoque geográfico.

1.7.2. Arquitectura del sistema

En el flujo de trabajo de la presente tesis, ilustrado en la Figura 1.16, se destacan dos herramientas interconectadas como parte central de la solución propuesta: el RNA y el ACG. Como se ilustra en la Figura 1.17, existen diferentes módulos a considerar en cada una de las dos herramientas. Los módulos en amarillo corresponden a las clases implementadas en base al estándar railML, los módulos en verde son aquellos algoritmos originales necesarios para resolver el problema, y finalmente los módulos en violeta constituyen todas las validaciones realizadas.

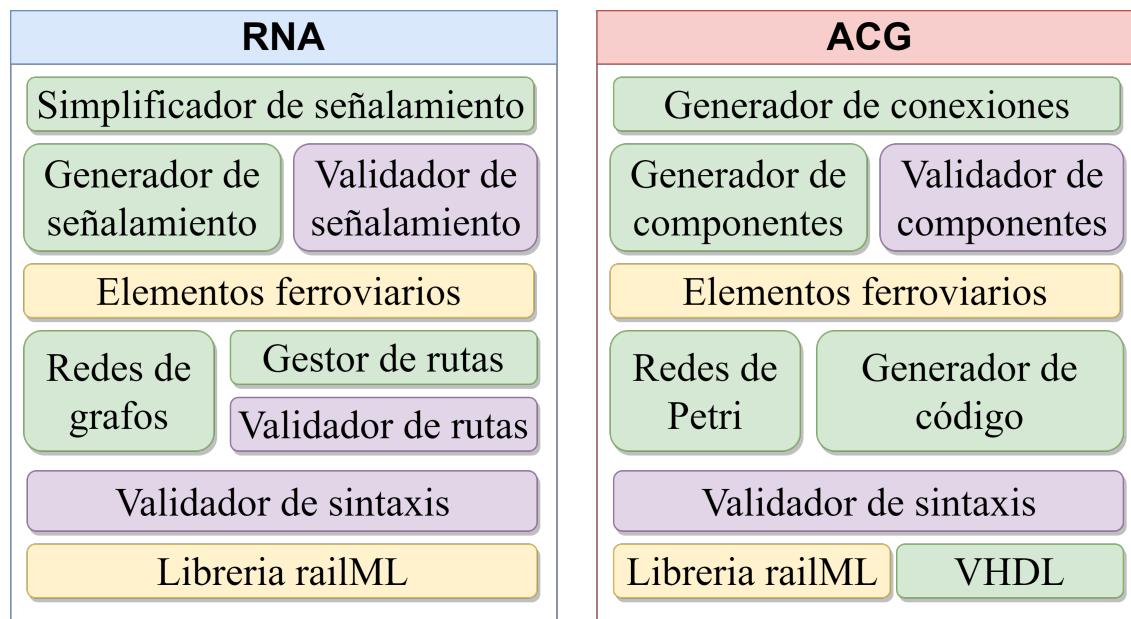


Figura 1.17: Arquitectura del sistema

Las clases implementadas se corresponden una a una con las definidas en el estándar railML. De esta manera, cualquier elemento definido en un archivo railML tendrá su equivalente en el modelo de datos utilizado. La biblioteca a implementar debe ser capaz de importar, modificar y exportar archivos en formato railML. Además, la librería debe contemplar todos los atributos que railML defina para cada elemento ferroviario.

Respecto al RNA, el análisis de redes de grafos es explicado en la Sección 2.1.3, la generación de señalamiento en la Sección 2.2, la simplificación del señalamiento en la Sección H.3 y, finalmente,

la generación de rutas ferroviarias y la tabla de enclavamientos es explicado en la Sección 2.4. En tanto que para el ACG, la descripción de cada bloque implementado se encuentra explicada en la Sección 3.2, la implementación de la redundancia en la Sección 3.3 y la plataforma utilizada se explica en la Sección 3.4.

1.7.3. Amenazas a la validez del sistema

Para asegurar la validez de nuestro enfoque, analizamos tanto la validez interna como la externa [15, 208, 209]. En cuanto a la validez interna, es importante garantizar una relación causal entre el diseño proporcionado y la señalización generada sin que ningún otro factor externo afecte el resultado. Podemos aprovechar el hecho de que el modelo ferroviario que el RNA utiliza es un sistema lineal debido a que el RNA se basa en redes de grafos [128, 130, 131, 133, 210] y las redes de grafos son sistemas lineales [59, 60, 63, 64, 108, 153, 160, 161, 173, 176, 180, 189, 190, 211-216]. De esta manera, las señales generadas para dos elementos ferroviarios A y B serían las mismas que las señales generadas para el elemento ferroviario A más las señales generadas para el elemento ferroviario B. Este análisis se puede extender a cualquier número de elementos ferroviarios diferentes. Esto puede provocar solapamientos de señales, por lo tanto, se añade la posibilidad de que los usuarios de la herramienta habiliten o deshabiliten la simplificación de las mismas, así como establecer la distancia mínima entre elementos para considerar si se superponen o no.

Los usuarios pueden seleccionar qué elementos ferroviarios se analizarán. La elección de los elementos a analizar es extremadamente poderosa porque nos permite probar la generación de señales para cada elemento ferroviario individualmente para validar que se estén analizando correctamente. Una vez que esta validación se realiza para cada elemento y se confirma que no hay factores externos que afecten la señalización generada, podemos seleccionarlos simultáneamente, sin simplificación. De esta manera, la linealidad del sistema se manifestará por sí misma y, en algunos casos, también la necesidad de simplificación de la señalización (ver Sección 2.3.3). Debido a la linealidad, los elementos ferroviarios que están muy cerca uno del otro se pueden considerar como un solo elemento, lo que resulta en una nueva señalización que tendría menos señales que la suma de sus señales anteriores debido a la simplificación de señales redundantes.

Respecto a la demostración formal de los resultados, tal como se mencionó en la Sección 1.6.2, los estudios académicos suelen depender en su mayoría de métodos formales para verificar y validar sus diseños. Sin embargo, el RNA se basa en redes de grafos y, según la norma IEC 61508 [67-73, 217], las redes de grafos son un método semi-formal. Además, casi la mitad de los estudios respaldados por las empresas más importantes de la industria ferroviaria utilizan técnicas semi-formales [218-229] para reducir la brecha entre la formalidad académica y las necesidades de la industria. Con el fin de realizar pruebas en sistemas ferroviarios reales, damos mayor énfasis a la validación de un enfoque más práctico y menos formal, de acuerdo con los requisitos y tendencias de la industria ferroviaria [67, 79-85, 88, 230-240].

Adicionalmente a la validación del resultado, es importante validar que el archivo railML generado sea sintácticamente correcto según los estándares de railML. El RNA realiza validaciones de sintaxis tanto al importar el archivo en formato railML como al generar uno nuevo. Esto se puede confirmar fácilmente importando el archivo railML en Design4Rail [146], una herramienta certificada por railML.org. Esta herramienta realiza una validación de sintaxis de nuestros archivos railML generados y también se utiliza para visualizar los diseños ferroviarios generados automáticamente.

En cuanto a la validez externa, la inclusión de nueve casos de estudio ferroviarios cuidadosamente seleccionados presentados en el Capítulo 4 tiene como objetivo cubrir una amplia gama de topologías y un uso extensivo de los elementos ferroviarios más comunes. Cuatro de estos ejemplos son diseños reales de ferrocarriles y los otros cinco se crearon para introducir topologías que se utilizan ampliamente en muchos países. Por lo tanto, cualquier otro diseño ferroviario compartiría la mayoría de las características y elementos modelados en nuestros ejemplos. Si se detecta un nuevo elemento en el diseño, se consideraría como un elemento a proteger y RNA generará la señalización

adecuada de acuerdo con los principios P3 y P6 explicados en la Sección 1.2.

En definitiva, nuestro proceso de validación implica una comparación automática entre las tablas de enclavamiento aprobadas por las autoridades ferroviarias y las tablas generadas por RNA. La ruta definida por RNA (o conjunto de rutas) debe tener una ruta correspondiente en el diseño original de señalización. Cualquier ruta no definida originalmente debe mejorar la seguridad general al proteger la infraestructura ferroviaria que no se consideró originalmente o mejorar la logística al agregar nuevas rutas no conflictivas. RNA también realiza una validación de sintaxis del archivo railML, indicando si alguna parte de la estructura XML está ausente o dañada.

Finalmente, validamos que todos los principios de señalización introducidos en este artículo sean cumplidos por el señalamiento generado por el RNA mediante un proceso de análisis posterior del diseño y la nueva señalización, asegurando que se cumpla cada principio de diseño ferroviario.

Capítulo 2

Railway Network Analyzer (RNA)

El sistema ferroviario consta de diversos elementos que incluyen la infraestructura, sensores, actuadores e interfaces visuales con los conductores. Todos estos elementos se interrelacionan y funcionan en conjunto dentro del sistema de señalamiento y el sistema de enclavamiento. Cada uno de estos elementos y cómo se relacionan se encuentra definido en el formato railML [128, 131-135].

El RNA deberá procesar cualquier archivo compatible con railML, detectar cada elemento junto a sus propiedades y cómo están conectados dentro de la red ferroviaria. En base a esta información, que define la infraestructura, es decir, las características estáticas de la red, el RNA deberá detectar los puntos críticos, o sea, las zonas en las cuales es mas probable que ocurran colisiones entre formaciones, ya sea de frente como lateralmente, así como también los descarrillamientos. Cada elemento ferroviario, según corresponda a cada elemento, será protegido por una serie de semáforos. La conjunción de todos estos semáforos probablemente dará lugar a superposiciones o contradicciones que luego serán simplificadas por el RNA antes de dar su resultado final.

En base al señalamiento generado, el RNA debe detectar todas las rutas posibles y generar una nueva tabla de enclavamientos. De existir un señalamiento previo, deberá compararlos y verificar que todas las rutas originales son equivalentes a alguna de las rutas nuevas. De lo contrario, deberá buscar combinaciones entre dos o mas rutas nueva que sean equivalentes a las rutas originales. Adicionalmente, el RNA validará que el señalamiento cumple los principios de señalamiento ferroviario establecidos en la Sección 1.2 y que la sintaxis del archivo railML obtenido es correcta.

2.1. Biblioteca RailML

El primer paso en el desarrollo del RNA es contar con una biblioteca que permita generar un objeto equivalente uno a uno con el estándar railML y luego exportar esa información en un nuevo archivo railML. Como ya se mencionó en la Sección 1.5, el estándar railML define cinco clases de objetos principales: *Common*, *Infrastructure*, *Interlocking*, *RollingStock* y *Timetable*.

La clase *Common* contiene 109 clases, todas ellas utilizadas para definir características generales e invariantes para la red. Por ejemplo, podemos encontrar la clase *Concessionaire* que define quien tiene la concesión de la red, la clase *tBrakeType* que define el tipo de frenos aceptados en esa red, la clase *tVoltage* que define la tensión de la red o la clase *PublicHolidayPeriodRule* que define el período de vacaciones de los operarios de la red, entre otros factores. Esta clase es obligatoria para todos los archivos en formato railML, por lo tanto es necesaria para el funcionamiento del RNA, aunque no se extraigan datos de la misma.

La clase *Infrastructure* incluye 231 clases, todas ellas enfocadas en definir las características estáticas de cada elemento ferroviario, como su posición, dimensiones y demás parámetros invariantes en el tiempo. Entre ellas podemos encontrar la clase *Tracks* (ver Sección 2.1.4), *Borders*

(ver Sección 2.1.5), *BufferStops* (ver Sección 2.1.5), *LevelCrossingsIS* (ver Sección 2.1.8), *TrackDetectionElement* (ver Sección 2.1.6), *Platforms* (ver Sección 2.1.7), *SwitchesIS* (ver Sección 2.1.9) y *SignalsIS* (ver Sección 2.1.10), entre otras. Adicionalmente incluye dos clases fundamentales para el análisis de redes de grafos: *netElement* y *netRelation*, ambas discutidas en la Sección 2.1.3. En el caso de hacer análisis a nivel "mesoscópico" "macroscópico" se tienen, además, las clases *Line*, *OperationalPoint* y *Network*, no cubiertas en este trabajo, ya que la información obtenida en estos niveles es redundante con la obtenida a nivel "microscópico".

La clase *Interlocking* incluye 156 clases, todas ellas enfocadas en definir las características dinámicas de los elementos ferroviarios que las posean. Entre ellas podemos encontrar *LevelCrossingsIL* (ver Sección 2.1.8) y *SwitchesIL* (ver Sección 2.1.9), necesarias tanto para el RNA como para el ACG. Adicionalmente, se encuentran las clases relacionadas a la tabla de enclavamientos y las rutas, como *Routes*, *CombinedRoutes*, *ConflictingRoutes*, *RoutesRelations*, y muchas otras referidas explícitamente a rutas, como *SignalsIL*, donde se definen perfiles de velocidad para las señales.

La clase *RollingStocks* incluye 30 clases, todas ellas enfocadas en definir las características de cualquier tipo de material rodante. Esta clase se divide en dos grandes clases: *Vehicle* y *Formations*. La primer clase contiene la información relacionada a vehículos unitarios tales como: vagón de carga, coche de pasajeros o una locomotora. La segunda clase combina los elementos de la clase *Vehicle* para definir las formaciones en toda su extensión.

La clase *TimeTables* incluye 148 clases, todas ellas enfocadas a la logística ferroviaria. Entre sus clases podemos encontrar información respecto al tipo de formaciones que circulan en la red (clases *CommercialTrains*, *OperationalTrains*, *TransportServices* y *Categories*), itinerarios (clases *ItinerariesTT*, *ConnectionTransferTimes*, *CommercialSchedulings*, *BaseItineraries* y *TimetableScenarios*) y los anuncios a los usuarios (clases *PassengerTextInfos* y *Announcements*).

El RNA solamente necesita de las clases *Common*, *Infraestructure* e *Interlocking* para funcionar, por lo que no fue necesario implementar las clases *RollingStock* ni *Timetable*. En total se han implementado 524 clases [241], considerando 28 clases nativas de RailTopoModel necesarias para que railML pueda interpretar los tipos de datos definidos. Todas las clases y clases mencionadas son opcionales en el estándar, salvo que se indique lo contrario, es decir, no son obligatorias para que el archivo railML se considere válido.

En las siguientes secciones se detallaran las clases mas importantes implementadas, tal que se pueda comprender que atributos son procesados para poder generar el señalamiento. Se comenzará con las clases relativas a la topología de la red, eje central del análisis del RNA y se proseguirá con las clases asociadas a elementos ferroviarios materiales. Simultáneamente, se profundizará en los conceptos mas destacados de cada elemento ferroviario correspondiente a esa clase, destacando cual es el señalamiento asociado al mismo para protegerlo en base a los principios ferroviarios expuestos.

2.1.1. Clase *Metadata*

Aunque no es una clase de railML, la clase *Metadata* es fundamental para que el archivo en formato railML sea válido. Se presenta el Código 2.1 para ilustrar los elementos presentes en la clase metadata.

Código 2.1: Clase *Metadata*

```

1 <metadata>
2   <dc:title>Example_9.railml</dc:title>
3   <dc:date>2023-10-04T10:51:21Z</dc:date>
4   <dc:creator>Trenes_Argentinos</dc:creator>
5   <dc:source>RaIL-AiD</dc:source>
6   <dc:identifier>1</dc:identifier>
```

```

7  <dc:subject>railML.org</dc:subject>
8  <dc:format>0.9.5</dc:format>
9  <dc:description>Ejemplo_real</dc:description>
10 <dc:publisher>RAIL-AiD framework</dc:publisher>
11 </metadata>

```

Ninguno de estos campos es esencial para el análisis de la red ferroviaria, por lo que son copiados sin cambios al nuevo archivo generado. No obstante, la ausencia de alguno de estos campos o que el campo sea nulo provoca que tanto el RNA como cualquier herramienta compatible con railML considere al archivo como incompleto o corrupto.

2.1.2. Clase *Common*

La clase *Common* define todos los parámetros que son invariantes a toda la red. Esta clase puede ser definida solo una vez o no definirse. Sus clases son:

- *ElectrificationSystems*: define la tensión y frecuencia de la red eléctrica.
- *OrganizationalUnits*: define quien administra la infraestructura, quién fabrica y opera el material rodante, quién es el cliente final, quién es el contratista y quién posee la concesión del servicio.
- *SpeedProfiles*: define el perfil de aceleración, velocidades y frenado.
- *Positioning*: define los sistemas de posicionamiento geométrico, lineal y referido a la pantalla del editor.

Cada uno de estos campos internos es único, pero también son opcionales. A modo de ejemplo se muestra el Código 2.2, donde se ilustra como no todas las clases han sido definidas.

Código 2.2: Clase *Common*

```

1  <common id="co_01">
2      <organizationalUnits>
3          <infrastructureManager id="im_01"/>
4      </organizationalUnits>
5      <positioning>
6          <geometricPositioningSystems>
7              <geometricPositioningSystem id="gps01">
8                  <isValid from="2023-07-26" to="2024-07-26"/>
9                  <name name="Example_9.railml" language="en"/>
10             </geometricPositioningSystem>
11         </geometricPositioningSystems>
12         <linearPositioningSystems>
13             <linearPositioningSystem linearReferencingMethod="absolute"
14                 startMeasure="0" endMeasure="0" units="Km" id="loc-1">
15                 <isValid from="2023-07-26" to="2024-07-26"/>
16                 <name name="Loc-1" language="en"/>
17             </linearPositioningSystem>
18         </linearPositioningSystems>
19     </positioning>
20 </common>

```

Solamente las clases *OrganizationalUnits* y *Positioning* fueron definidas, pero tanto el RNA como el estándar railML en el que el RNA se basa consideran válido a este fragmento de código. Los vectores son definidos en plural, como en el caso de *geometricPositioningSystems* cuyo primer, y único elemento en este caso, es *geometricPositioningSystem* con id="gps01". Es habitual ver estos vectores a lo largo de todo el archivo y será esencial poder determinar su dimensión para procesar correctamente los datos y contar la cantidad de elementos.

2.1.3. Clase *Infraestructure* y redes de grafos

La clase *Infraestructure* define todos los elementos ferroviarios con características físicas, materiales y estáticas. Las clases que contiene son:

- *Topology*: define la topología de la red mediante las clases *netElements* y *netRelationships*. Incluye también la clase *Network* para cumplir con el estándar de RailTopoModel.
- *Geometry*: define el sistema geométrico utilizado entre las clases *HorizontalCurve* (en base al largo y el ángulo formado), *GradientCurves* (en base al largo y al gradiente de la curva) y *GeometryPoints* (en base a una combinación de las dos clases anteriores).
- *FunctionalInfrastructure*: define todos los elementos ferroviarios que el RNA analizará.
- *PhysicalFacilities*: actualmente el estándar lo define vacío, sin ninguna clase interna salvo la clase *Any* que puede ser utilizada como comodín.
- *InfrastructureVisualizations*: define las coordenadas y estructura del elemento ferroviario en base a sus clases *tRef* (para indicar a qué elemento afecta), *SpotProjection* (coordenada del elemento), *LinearProjection* (en caso de ser un elemento lineal) y *AreaProjection* (en caso de ser un elemento bidimensional).
- *InfrastructureStates*: define la validez de los datos referidos a cada elemento ferroviario.

Para realizar el análisis de la red, el RNA debe centrarse en tres de estas clases: *Topology* (para conocer cómo es la red), *FunctionalInfrastructure* (para conocer qué elementos tiene la red) y *InfrastructureVisualizations* (para conocer dónde está cada elemento ferroviario).

Empezando por la clase *Topology*, existen dos clases esenciales para este análisis: la clase *netElements* y la clase *netRelationships*. Ambas son clases vectores, constituidas por clases más pequeñas: *netElement* y *netRelationship*. Un ejemplo de la clase *netElement* se puede ver en el Código 2.3.

Código 2.3: Clase *netElement*

```

1 <netElement id="ne3">
2   <associatedPositioningSystem id="ne3_aps01">
3     <intrinsicCoordinate intrinsicCoord="0" id="ne3_aps01_ic01">
4       <geometricCoordinate x="9384.050" y="0.000" positioningSystemRef="gps01"/>
5     </intrinsicCoordinate>
6     <intrinsicCoordinate intrinsicCoord="1" id="ne3_aps01_ic02">
7       <geometricCoordinate x="7584.770" y="0.000" positioningSystemRef="gps01"/>
8     </intrinsicCoordinate>
9   </associatedPositioningSystem>
10  <relation ref="nr_ne3ne46_swi77"/>
11  <relation ref="nr_ne3ne53_swi77"/>
12 </netElement>

```

Los elementos mas importantes a destacar en el Código 2.3 son el *id*, el *geometricCoordinate* y el *relation*. De estos parametros podemos saber que el *netElement* es referido como "ne3", comienza en la coordenada (9384.050 ; 0.000) y termina en la coordenada (7584.770 ; 0.000). Además, ne3 se encuentra relacionado a los *netElement* ne46 y ne53 mediante un elemento referido como swi77, que más adelante veremos que se trata de un cambio de vías.

Los *netElement* son los nodos de la red de grafos, pero estos no son puntuales, sino que son bidimensionales. Debido a que la componente x de la coordenada de inicio de ne3 es mayor que la componente x de la coordenada final de ne3, podemos deducir que ne3 se encuentra definido de derecha a izquierda. Conocer la orientación del *netElement* será importante a la hora de definir la circulación, información necesaria para generar las rutas.

La clase *netRelation* son las aristas de la red de grafos que relacionan dos *netElements* entre sí. En el Código 2.4 se muestra un ejemplo de los *netRelation* que vinculan a los *netElement* ne3, ne46 y ne53.

Código 2.4: Clase *netRelation*

```

1 <netRelation navigability="Both" positionOnA="1" positionOnB="1" id="nr_ne3ne46_swi77">
2   <elementA ref="ne3"/>
3   <elementB ref="ne46"/>
4 </netRelation>
5 <netRelation navigability="Both" positionOnA="1" positionOnB="1" id="nr_ne3ne53_swi77">
6   <elementA ref="ne3"/>
7   <elementB ref="ne53"/>
8 </netRelation>
9 <netRelation navigability="None" positionOnA="1" positionOnB="1" id="nr_ne46ne53_swi77">
10  <elementA ref="ne46"/>
11  <elementB ref="ne53"/>
12 </netRelation>
```

La primer *netRelation* es entre el *netElement* ne3 y el ne46, la segunda *netRelation* es entre el *netElement* ne3 y el ne53. Esto es consistente con los parámetros de relation que tenía el *netElement* ne3 en el Código 2.3. Sin embargo, en el tercer *netRelation* vemos que existe una relación entre los *netElement* ne46 y ne53 pero, en este caso, el parámetro *navigability* es "None", a diferencia de los primeros dos que era "both". La navegabilidad es el parámetro que hace que una red de grafos tenga sentido como red ferroviaria: que exista una conexión no implica que sea físicamente utilizable por un tren, tal como se explicó en la Sección 1.5.1.2.

La secuencia de análisis de la red de grafos se describe en el Algoritmo 1. Tanto el Algoritmo 1 como todos los algoritmos pertenecientes al RNA y el ACG fueron diseñados e implementados durante el transcurso del presente doctorado. Estos algoritmos no son parte de ningún estándar ni trabajo previo ajeno a este desarrollo, por lo que son una pieza fundamental y original de esta tesis.

Algoritmo 1: Análisis de la red de grafos

```

{nodes} ← get_nodes(netElements)
order_nodes({nodes})
{netPaths} ← get_relations({nodes},netRelations)
{neighbours} ← get_neighbours({netPaths})
{switches} ← get_switches({nodes},{neighbours})
{limits} ← get_limits({nodes})
analyze_connectedness({netPaths})
```

El primer paso es procesar la clase *netElements* para obtener un diccionario de todos los nodos de la red. Ordenar este diccionario es importante para agilizar la búsqueda de rutas más adelante.

El criterio de ordenamiento fue por coordenadas, de menor a mayor, priorizando la coordenada x por sobre la y, pero cualquier otro criterio sería válido. Con el diccionario de nodos y analizando la clase *netRelations* es posible construir un diccionario de *netPaths*. Este diccionario utiliza cada nodo del grafo como índice y contiene un diccionario con todos los nodos que son vecinos anteriores y posteriores (considerando el criterio de ordenamiento, algunos nodos estarán antes o después).

La cantidad de vecinos de cada nodo es fundamental para poder clasificarlos. Los nodos que poseen un solo vecino serán nodos que se encuentren en el límite de la red (límite relativo o absoluto). Aquellos con dos vecinos, uno anterior y otro posterior, son nodos normales. En cambio, si tienen dos vecinos y ambos son anteriores o ambos son posteriores entonces son nodos que con toda seguridad tienen un cambio de vías. Por ejemplo, el nodo ne3 del Código 2.4 posee un cambio de vías ya que los nodos ne46 y ne53 son anteriores a ne3.

A continuación, es necesario determinar la conexidad de la red. Una red de grafos puede ser como alguna de las ilustradas en la Figura 2.1. Las redes de grafos que tengan nodos aislados o una cantidad de vecinos tales que ese nodo no represente a ningún cambio de vías existente no podrán ser consideradas una red ferroviaria. La red no tiene que ser totalmente conexa, se admiten redes ferroviarias disjuntas, siempre que cada red funcione de manera independiente, con una cantidad mínima de dos nodos en cada una.

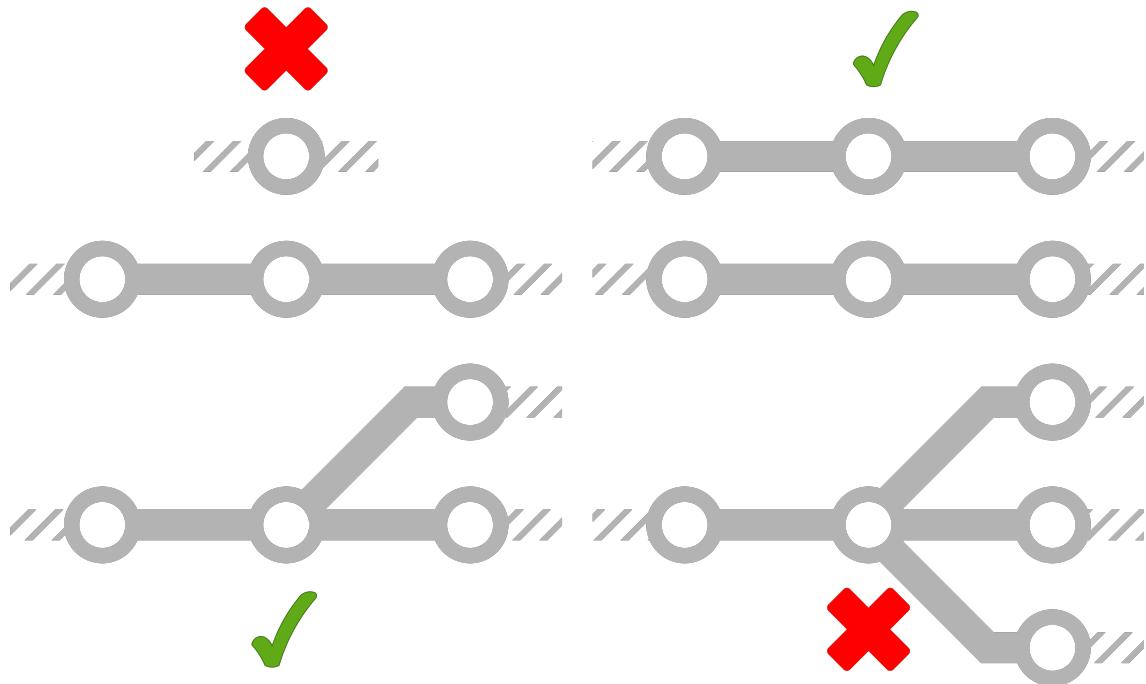


Figura 2.1: Distintos tipos de redes de grafos.

Para determinar si la red es conexa, se siguen los pasos del Algoritmo 2, y así se añaden nodos a una lista de zonas siempre que el nodo tenga un vecino en esa lista. Si un nodo posee un vecino en ambas listas, las listas se combinan, creando una nueva zona. Si la cantidad de zonas es uno entonces la red es fuertemente conexa. Una red ferroviaria desconexa puede resolverse más rápido que una fuertemente conexa ya que no haría falta iterar entre todos los elementos para analizarlos sino solo los pertenecientes a dicho subgrafo.

Algoritmo 2: Algoritmo de conexidad

```

{zones} ← first node
for node in {nodes} do
    for zone in {zones} do
        if node not in zones[zone] then
            if neighbours(node) in zones[zone] then
                zones[zone] ADD node
            else
                Define new_zone
                zones[new_zone] ADD node
            end if
        end if
    end for
end for
Return {zones}

```

En las siguientes subsecciones se analizaran los elementos ferroviario descriptos en las clases *FunctionalInfrastructure* y *InfrastructureVisualizations*.

2.1.4. Vías

Las vías férreas son el elemento ferroviario mas esencial, son la columna vertebral de la infraestructura ferroviaria. Estas constituyen el sitio por el cual se desplazan los trenes, definiendo no solo la dirección del desplazamiento, sino también restringiendo el dominio del tren. Esto lo diferencia de otros medios de transporte como el automóvil que, aún teniendo una carretera, puede moverse por fuera de esta.

Las vías se encuentran separadas por una distancia fija que se mide desde sus caras internas, denominada trocha (Figura 2.2). Solamente las formaciones compatibles con ese parámetro de trocha pueden circular por el tendido ferroviario. El valor de la trocha puede variar entre las denominadas trocha angosta (600 a 1372 mm, estándar imperial británico) y trocha ancha (1520 a 2140 mm, estándar ruso, indio, ibérico, irlandés). Se estableció el valor intermedio de 1435 mm como valor de trocha internacional, usado ampliamente en Europa, Norteamérica y Oceanía.

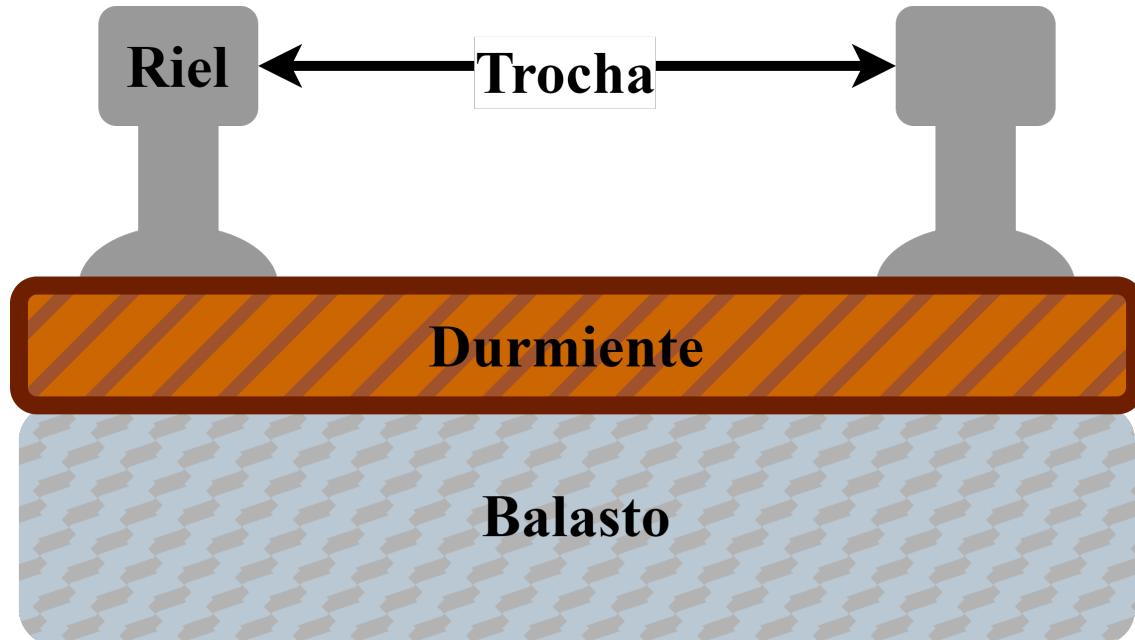


Figura 2.2: Vías ferroviarias y trocha.

Existen limitaciones logísticas y físicas por las cuales el tendido ferroviario no puede ser un continuo rígido. En primer lugar, las vías deben ser de un tamaño acotado, tal que puedan transportarse a la locación donde serán instaladas en tramos rectos o curvos. En segundo lugar, la dilatación y contracción de las vías debido a los cambios de temperatura añaden una restricción respecto a la distancia mínima que deben tener entre las mismas. De lo contrario, la dilatación del material puede provocar daños irreparables a la infraestructura y estos, a su vez, ser motivo de descarrilamientos, como ya ha ocurrido en los comienzos de la industria ferroviaria [242].

Cada vía puede ser clasificada en dos grupos: vías ascendentes o vías descendentes (ver Figura 2.3). Las ascendentes son aquellas por las cuales los trenes circulan únicamente en la dirección del kilometraje en sentido creciente. Las descendentes son aquellas por las cuales los trenes circulan únicamente en la dirección del kilometraje en sentido decreciente [10].

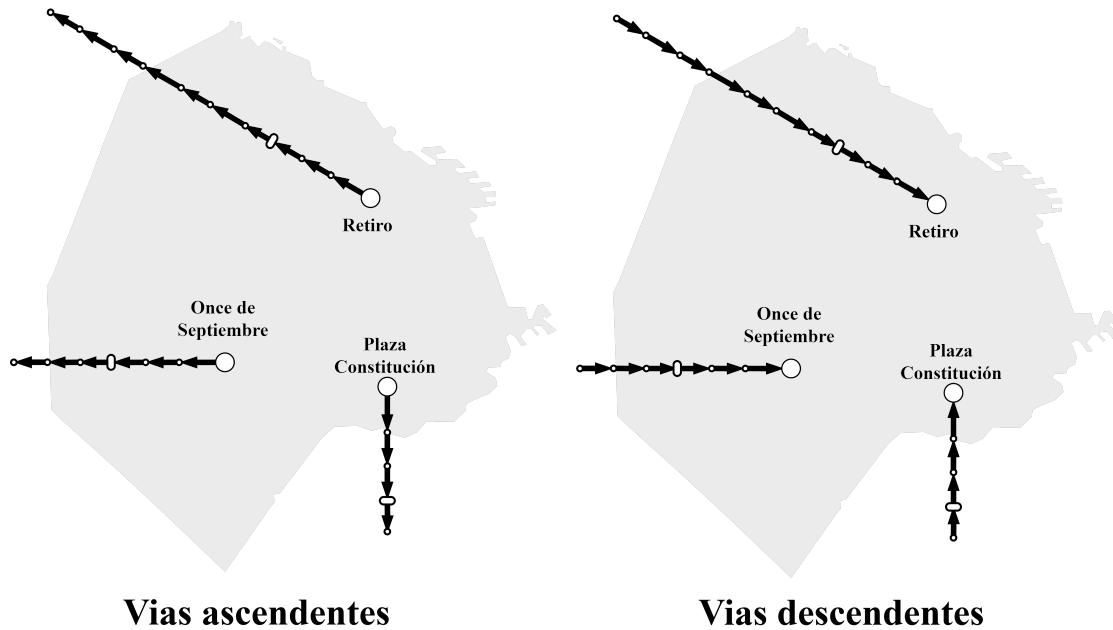


Figura 2.3: Vías ascendentes y descendentes.

El kilómetro cero es la estación principal de la línea ferroviaria, como pueden ser las terminales de Plaza Constitución (para la línea Roca), Once de septiembre (para la línea Sarmiento) y 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.

Las vías se agrupan en secciones que, por cuestiones de seguridad y logística, se establece que solo pueden ser utilizadas por un tren a la vez. Estas secciones pueden ser de varios kilómetros en zonas rurales o unos pocos cientos de metros en zonas urbanas, donde la red necesita una mayor granularidad debido a la densidad del tráfico ferroviario en las grandes urbes.

En railML las vías son elementos ferroviarios físicos, representados por la clase *track*, ilustrada en el Código 2.5. Esta clase se encuentra definida dentro del vector de clases *tracks*, dentro de la clase *functionalInfrastructure*, dentro de la clase *infrastructure*.

Código 2.5: Clase *Track*

```

1  <track type="mainTrack" infrastructureManagerRef="im_01" mainDirection="both" id="trk2"
2   ↵ " >
3   <trackBegin ref="bus5"/>
4   <trackEnd ref="swi77"/>
5   <length value="1799.28" type="physical"/>
6   <designator register="Example" entry="TRACK

```

Las vías en railML se definen entre dos elementos ferroviarios físicos. En este caso, la vía indicada como trk2 se encuentra definida entre el *BufferStop* bus5 (ver Sección 2.1.5) y el cambio

de vías swi77 (ver Sección 2.1.9). Esta clase define, además, el largo de la vía y el *netElement* al cual están asociadas, en este caso, el *netElement* ne3. Es natural confundirse *tracks* y *netElements*, porque son términos casi equivalentes, pero un track representa un elemento físico y un *netElement* engloba de manera abstracta una porción del trazado de vías. Es decir, una vía puede dividirse en varios *netElements*, pero un *netElement* solo se asocia a una vía.

2.1.5. Fin de vía y transiciones

El tendido ferroviario no puede continuar de forma indefinida, ni tampoco interrumpirse de forma abrupta. Como se puede visualizar en la Figura 2.4, existen dos formas de definir el fin de vía: de forma relativa y de forma absoluta.

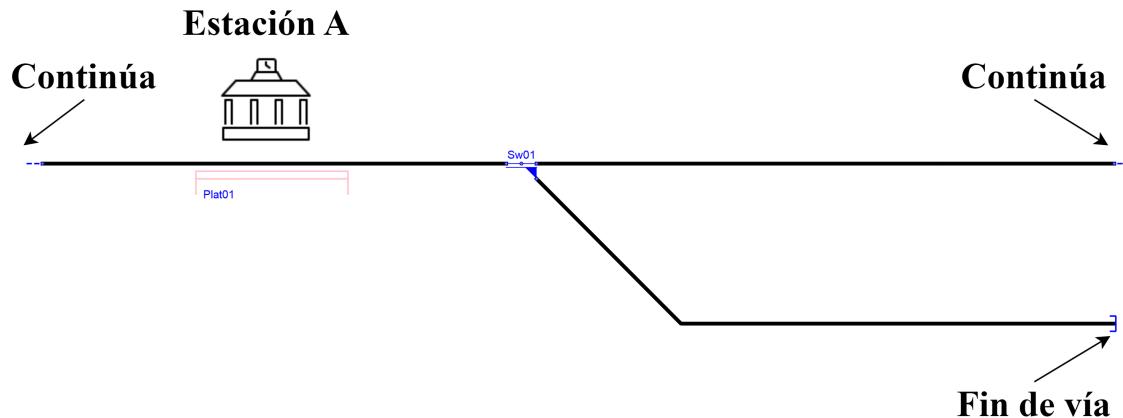


Figura 2.4: Topología de ejemplo con finales de vía relativos y absolutos.

Un ejemplo de fin de vía relativo sería las fronteras entre la estación A y la continuación de la vía principal. Las vías continúan a la izquierda y derecha de A, pero no son parte del alcance del sistema que controla los elementos ferroviarios en las inmediaciones de A. En tanto que un fin de vía absoluto sería todo final literal de la red ferroviaria, tales como terminales, talleres y vías secundarias sin retorno, como la ramificación saliente de la vía principal.

Un fin de vía relativo es todo aquel límite que define el alcance del sistema de señalamiento, pero no necesariamente es el final de la red. Es decir, el sistema podrá recibir información de cualquier sensor dentro de estos límites, o incluso de algunos sensores en la frontera exterior, pero solo podrá comandar los actuadores dentro de la zona. Esto es de vital importancia a la hora de reducir la complejidad de la red, al realizar el procesamiento de forma distribuida, sin depender de un decisión centralizada.

Este elemento ferroviario es modelo por la clase *border*, cuyo ejemplo se visualiza en el Código 2.6. Siempre se definen referido a un único *netElement*, en este caso el *netElement* ne114 y a una coordenada intrínseca que puede ser 0 si se encuentra al principio del *netElement*, o de 1 si se encuentra al final.

Código 2.6: Clase *Border*

```

1 <border isOpenEnd="false" externalRef="" type="station" id="sb540">
2   <designator register="Example" entry="BORDER_SB05"/>
3   <spotLocation netElementRef="ne114" intrinsicCoord="0.0000" applicationDirection=""
4     ↵ reverse" id="sb540_sloc01"/>
      <name name="SB05" language="en"/>
```

5 </border>

Un fin de vía absoluto es todo aquel límite que define el final físico del tendido ferroviario, luego del cual ya no se tiene infraestructura. Podemos encontrar estos límites en los talleres ferroviarios, en las terminales principales de cada red o en vías utilizadas para estacionar los trenes por fuera de la rama principal en uso. Estos límites deben estar adecuadamente protegidos y señalizados.

Este elemento ferroviario es modelo por la clase *bufferStop*, cuyo ejemplo se visualiza en el Código 2.7. Al igual que la clase *border*, se definen referido a un único *netElement*, en este caso el *netElement ne3* y a una coordenada intrínseca que puede ser 0 si se encuentra al principio del *netElement*, o de 1 si se encuentra al final.

Código 2.7: Clase *BufferStop*

```

1  <bufferStop type="fixedBufferStop" id="bus5">
2    <designator register="Example" entry="BUFFERSTOP_Buf05"/>
3    <spotLocation netElementRef="ne3" intrinsicCoord="0.0000" applicationDirection="
4      ↵ reverse" id="bus5_sloc01"/>
5    <name name="Buf05" language="en"/>
</bufferStop>

```

2.1.6. Sistemas de detección de formaciones ferroviarias

Es de vital importancia que el sistema pueda determinar la posición de un tren dentro del tendido ferroviario. De esta manera, poder habilitar la circulación en secciones donde no exista peligro de colisión con otros formaciones o, por el contrario, detener la marcha de las formaciones anteriores para evitar accidentes. Existen diversas maneras de detectar la posición de un tren, entre ellas el uso de circuitos de vía y contadores de ejes (Figura 2.5).



Figura 2.5: Circuito de vía (01) y contador de ejes (AxC01).

Los circuitos de vía (Figura 2.7) son dispositivos eléctricos que aplican una diferencia de potencial entre los rieles. Cuando una formación ingresa a la sección, sus ruedas metálicas cortocircuitan ambos rieles. El cortocircuito es detectado por el relé, que a su vez, reporta el estado al resto del sistema.



Figura 2.6: Circuito de vía libre y ocupado.

En caso de que la alimentación se interrumpa, el cableado sufra alguna falla, vandalismo, inundación, o que efectivamente una formación ocupe la sección, el circuito de vía reportará que la sección se encuentra ocupada. De esta manera, solo es posible recibir un reporte de sección desocupada cuando la sección efectivamente se encuentre desocupada. A este principio se le denomina fail-safe [222, 223, 243, 244]. Es decir, si por alguna razón algo falla, el sistema adopta la condición más restrictiva, mitigando la posibilidad de una situación peligrosa.

Existe una discontinuidad entre los rieles denominado juntura, que permite la expansión de los mismos al ser sometidos a altas temperaturas sin que los rieles se doblen y provoquen daños a la infraestructura. Los circuitos de vía, se relacionan directamente a las junturas entre las vías, modelados por la clase *railJoint* en railML. Esta discontinuidad eléctrica es lo que limita la acción del circuito de vía a la región entre dos junturas.

En el código 2.8 podemos ver un ejemplo de la clase *tvdSection* dentro de la clase *interlocking*, que incluye a la clase *assetsForIL* y estos, a su vez, a la clase vector de *tvdSections*. Esta clase posee un id (*tvd7*), la tecnología utilizada (en este caso *trackCircuit*, circuito de vía) y los límites donde el circuito de vía es válido: los *bufferStop* bus1 y bus2.

Código 2.8: Clase *TrackCircuit*

```

1 <tvdSection id="tvd7" partialRouteReleaseDelay="PT4S" residualRouteCancellationDelay=
2   ↪ PT90S" technology="trackCircuit" isBerthingTrack="false">
3   <designator register="Example" entry="01"/>
4   <hasDemarcatingBufferstop ref="bus2"/>
5   <hasDemarcatingBufferstop ref="bus1"/>
</tvdSection>
```

Los sistemas contadores de ejes (Figura 2.7) consisten en sensores pasivos instalados en la cara interna de unos de los rieles y un sistema externo de procesamiento de datos. Estos sistemas no dependen de la aplicación de tensiones en la vía. Además, no solo permiten detectar la presencia de una formación, sino que también pueden usarse para medir la integridad de la formación, si se conoce a priori la cantidad de ejes de la misma.



Figura 2.7: Contadores de ejes.

Al igual que los circuitos de vía, los sistemas contadores de eje siguen el principio de fail-safe, adoptando la condición mas restrictiva en caso de falla. Ambos sistemas pueden utilizarse en simultáneo, de ser requerido. En el código 2.9 podemos ver un ejemplo de la clase *trainDetectionElement*, dentro de la clase *functionalInfrastructure*, dentro de la clase *infrastructure*. En este ejemplo la bclase fue definida como tipo *axleCounter* (contador de eje) y se le asigna el nombre AxC01 que vemos en la Figura 2.5, referido al *netElement* ne1. También podemos ver que este contador de ejes se activa en ambos sentidos (*applicationDirection* = both) y su coordenada intrínseca dentro del *netElement* ne1, dentro de los dos tercios de la sección.

Código 2.9: Clase *TrainDetectionElement*

```

1 <trainDetectionElement id="ac6" type="axleCounter">
2   <name name="AxC01" language="en"/>
3   <spotLocation id="ac6_sloc01" netElementRef="ne1" applicationDirection="both"
4     ↪ intrinsicCoord="0.6710"/>
5   <designator register="Example" entry="TRAIN_DETECTION_ELEMENT_AxC01"/>
</trainDetectionElement>
```

2.1.7. Estaciones ferroviarias

Las estaciones ferroviarias son las zonas donde las formaciones pueden detenerse para que los pasajeros puedan descender y nuevos pasajeros puedan abordar. En función del tamaño de las formaciones y la geografía del lugar, las plataformas desde donde ascienden y descienden los pasajeros pueden estar elevadas con respecto al suelo o a ras del mismo. El largo de las plataformas también depende de la cantidad de coches de las formaciones.

Como puede verse en la Figura 2.8, las estaciones ferroviarias incluyen no solo a las plataformas, sino que también pueden centralizar el control de varias operaciones logísticas como la asignación de rutas. No obstante, en este trabajo nos referiremos a las estaciones como plataformas indistintamente.



Figura 2.8: Estación de doble plataforma.

Las estaciones de mayor complejidad o de mayor convergencia de ramales suelen concentrar el control de la estación donde se encuentran y varias estaciones vecinas. Las estaciones terminales, a menudo, pueden incluso tener control total de varios ramales completos.

En el Código 2.10 se puede ver la implementación de la clase *platform* de las plataformas ilustradas en la Figura 2.8. Esta clase se encuentra dentro de la clase *infrastructure*, que a su vez contiene a la clase *functionalInfrastructure*. La misma incluye atributos como la altura, el nombre asignado que se visualiza en la Figura 2.8 y el *netElement* al que se encuentran asociadas cada una de las plataformas.

Código 2.10: Clase *Platform*

```

1 <platforms>
2   <platform id="plf3" height="0">
3     <name name="Plat01" language="en"/>
4     <linearLocation id="plf3_lloc01" applicationDirection="both">
5       <associatedNetElement keepsOrientation="true" netElementRef="ne1"/>
6     </linearLocation>
7     <designator register="_Example" entry="PLATFORM_Plat01"/>
8     <length type="physical" value="0" validForDirection="both"/>
9   </platform>
10  <platform id="plf6" height="0">
11    <name name="Plat02" language="en"/>
12    <linearLocation id="plf6_lloc01" applicationDirection="both">
13      <associatedNetElement keepsOrientation="true" netElementRef="ne3"/>
14    </linearLocation>
15    <designator register="_Example" entry="PLATFORM_Plat02"/>
16    <length type="physical" value="0" validForDirection="both"/>
17  </platform>
18 </platforms>

```

Los datos relativos a la características físicas como el largo y el ancho la encontramos dentro de la clase *visualization*, dentro de la clase *infrastructureVisualizations*, como se puede ver en el Código 2.11.

Código 2.11: Clase *Visualization*

```

1 <spotElementProjection refersToElement="plf3" id="vis01_sep4">
2   <name name="Plat01" language="en"/>
3   <coordinate x="-125.455" y="-240.000"/>

```

```

4  </spotElementProjection>
5  <spotElementProjection refersToElement="plf6" id="vis01_sep5">
6    <name name="Plat02" language="en"/>
7    <coordinate x="-125.454" y="-30.000"/>
8  </spotElementProjection>

```

Conocer las dimensiones del elemento físico que representa esta clase es fundamental a la hora de analizar el impacto de este elemento en la red y donde se deberían asignar las señales ferroviarias.

2.1.8. Cruces ferroviarios

Los cruces ferroviarios son la intersección entre la vía ferroviaria y una ruta vehicular o peatonal. Estos cruces pueden ser bajo nivel (túnel por debajo de la vía), sobre nivel (puente vehicular por sobre la vía) o a nivel. Un paso a nivel es una zona muy crítica del sistema ferroviario, ya que, a diferencia de un paso sobre nivel o bajo nivel, conviven simultáneamente la formación y el transito vehicular y peatonal. En la Figura 2.9 se ilustra la intersección entre el tendido ferroviario, un cruce vehicular y un cruce peatonal.



Figura 2.9: Cruces de vías vehiculares y peatonales.

Los pasos a nivel peatonales incluyen un pequeño laberinto zigzagueante para forzar al peatón a aminorar su marcha y ver a ambos lados antes de cruzar las vías. A menudo suelen estar acompañados de indicaciones lumínicas y sonoras que se accionan tan pronto el tren se encuentre dentro de un rango de varios metros cercano al paso a nivel, representados por las secciones de vía coloreadas en amarillo.

Los pasos a nivel vehiculares añaden barreras ferroviarias para detener el tráfico vehicular cuando un tren se encuentra dentro de un rango de seguridad definido, representados por las secciones de vía coloreadas en rojo. El sistema de control de la barrera mantiene el brazo de esta en alto para permitir la circulación vehicular. Si un tren es detectado cerca del paso a nivel, se desenergiza la barrera y comienza a descender el brazo por efecto de la gravedad. 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 próximas al paso a nivel, la barrera vuelve a energizarse y se sitúa en estado alto nuevamente, a la espera de otro tren para reiniciar el proceso.

Se debe destacar que el mismo proceso de descenso de la barrera ocurrirá si esta se desenergiza por una falla electromecánica 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.

En el Código 2.12 se puede ver la implementación de la clase *levelCrossingsIS* que modela los cruces ferroviarios. Esta clase se encuentra dentro de la clase *infrastructure*, que a su vez contiene

a la clase *functionalInfrastructure*. La misma incluye atributos como el nombre asignado, si posee o no indicación lumínica o sonora, la dirección de activación y el *netElement* al que se encuentran asociado, además de su coordenada relativa dentro del *netElement*.

Código 2.12: Clase *LevelCrossingsIS*

```

1 <levelCrossingsIS>
2   <levelCrossingIS id="lcr7" activation="none">
3     <name name="Lc01" language="en"/>
4     <spotLocation id="lcr7_sloc01" netElementRef="ne1" applicationDirection="both"
5       ↪ intrinsicCoord="0.3145"/>
6     <designator register="_Example" entry="LEVEL_CROSSING_Lc01"/>
7     <protection barriers="none" lights="none" acoustic="none" hasActiveProtection="
8       ↪ true"/>
9   </levelCrossingIS>
10 </levelCrossingsIS>
```

Los datos relativos a la características físicas como el largo y el ancho la encontramos dentro de la clase *visualization*, dentro de la clase *infrastructureVisualizations*, como se puede ver en el Código 2.13. Conocer las dimensiones del elemento físico que representa esta clase es fundamental a la hora de analizar el impacto de este elemento en la red y donde se deberían asignar las señales ferroviarias.

Código 2.13: Clase *Visualization*

```

1 <spotElementProjection refersToElement="lcr7" id="vis01_sep4">
2   <name name="Lc01" language="en"/>
3   <coordinate x="-236.364" y="-240.000"/>
4 </spotElementProjection>
```

La clase *levelCrossingIL*, definida dentro del vector de clases *LevelCrossingsIL*, se encuentra dentro de la clase *assetsForIL*, en la clase *interlocking*. Contiene datos extra sobre el comportamiento dinámico de las barreras de los pasos a nivel y define los estados por defecto de la barrera. El RNA no hace uso de esta información, pero si el ACG.

Código 2.14: Clase *LevelCrossingIL*

```

1 <levelCrossingIL preferredPosition="closing" typicalTimeToClose="PT10S" id="
2   ↪ il_lcr176">
3   <isLevelCrossingType ref="lcx_aut"/>
4   <refersTo ref="lcr176"/>
5 </levelCrossingIL>
```

2.1.9. Cambios de vías

Un cambio de vías es un elemento ferroviario dinámico, que puede adoptar diferentes posiciones que modifican la topología de la red. Una formación por si sola no puede hacer mas que avanzar o retroceder, pero utilizando un cambio de vías es posible trazar nuevos caminos en el tendido ferroviario, modificando la posición de los cambios de vías.

Los cambios de vías pueden ser simples, dobles o en tijeras. Las posiciones que pueden adoptar se visualizan en la Figura 2.10. Un cambio de vías simple permite dos circulaciones posibles, una por vía principal y otra por vía secundaria. Un cambio de vías dobles permite cuatro circulaciones

posibles, todas de igual prioridad. Un cambio de vías en tijeras permite dos circulaciones posibles, ambas principales.

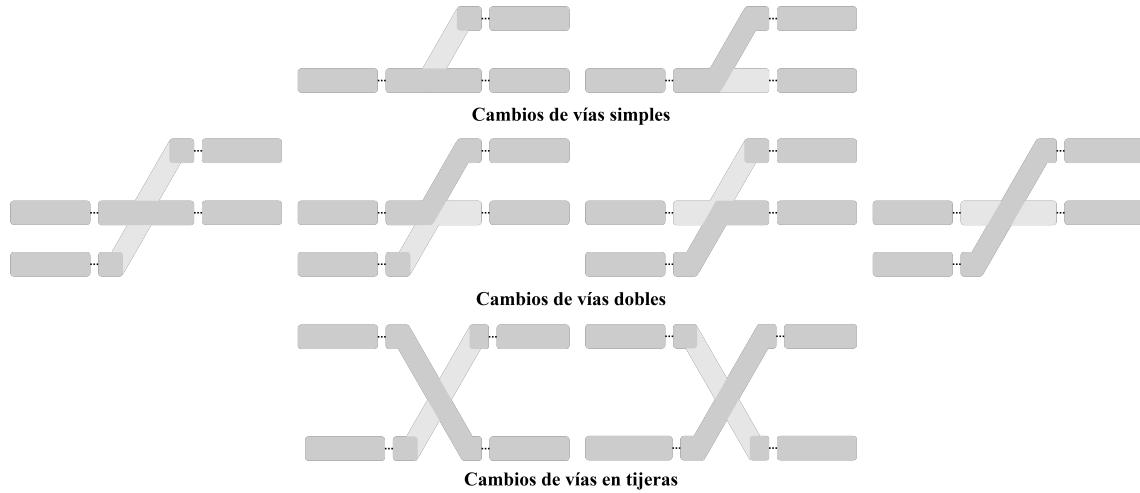


Figura 2.10: Circulaciones posibles en cambios de vías.

El sistema de enclavamientos modifica la posición de los cambios de vías por medio de una máquina de cambios. Una máquina de cambios (Figura 2.11) 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.



Figura 2.11: Máquina de cambios.

En la Figura 2.12 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 2.12: Cambio de vías de estación Matheu, Línea Mitre.

Al ser mecanismos que necesitan tiempo para cambiar de un estado al otro, no puede asumirse que el comando es obedecido al instante. Incluso podría darse el caso que jamás llegue a cumplirse la orden debido a desperfectos mecánicos o eléctricos. Es por eso que introducimos los conceptos de comando, indicación y correspondencia, tal como se ilustran en la Figura 2.13.



Figura 2.13: Comando, indicación y correspondencia en un cambio de vías.

El comando es la instrucción que el sistema de enclavamientos envía a la máquina de cambios. Esta instrucción puede ser modificar la posición de normal a reverso o de reverso a normal. La indicación es el estado que la máquina de cambios informa al sistema de enclavamientos. El sistema solo asume que el comando fue obedecido cuando tanto el comando como la indicación muestran una correspondencia. En caso contrario, el sistema de enclavamiento no puede asumir cual es el estado real del sistema, si el comando enviado o el estado reportado por la máquina de cambios. El mismo concepto puede ser aplicado en cualquier otro elemento mecánico, como por ejemplo las barreras ferroviarias.

El RNA debe analizar diversos atributos distribuidos entre la clase *switchIS* (Código 2.15 y Código 2.18), la clase *crossings* (Código 2.21), la clase *spotElementProjection* (Código 2.16, Código 2.19), la clase *switchIL* (Código 2.17 y Código 2.20) y la clase *movableCrossings* (Código 2.22).

La clase *switchIS*, que define a los cambios de vías simples y dobles, se encuentra dentro del vector de clases *switchesIS*, dentro de la clase *functionalInfrastructure*, que a su vez es parte de la clase *infrastructure*. La clase *switchIS* define el id del cambio de vías, el tipo (*ordinarySwitch* para cambio de vías simple o *doubleSwitchCrossing* para cambios de vías dobles), el *netElement* al cual pertenece la entrada del cambio y hacia que lado se encuentra la vía de continuación y ramificación si transitamos desde el *netElement* del cambio hacia el cambio mismo.

En el caso del cambio de vías simple (Código 2.15), si transitamos por el *netElement* ne16 el tren tendrá la vía de continuación en la mano derecha y la ramificación en la mano izquierda. Los cambios de vías simples (*ordinarySwitch*) siempre tienen una rama izquierda y una rama derecha definida. Además, dentro de la definición de cada rama tenemos el atributo *netRelationRef*, del cual se puede obtener, procesamiento mediante, los otros *netElement* correspondientes a las ramas: ne15 y ne14.

Código 2.15: Clase *switchIS* (*ordinarySwitch*)

```

1 <switchIS id="swi84" continueCourse="right" branchCourse="left" type="ordinarySwitch">
2   <name name="Sw01" language="en"/>
3   <spotLocation id="swi84_sloc01" netElementRef="ne16" applicationDirection="reverse"
4     ↪  intrinsicCoord="0.0000"/>
5   <designator register="_Example" entry="SWITCH\Sw01"/>
6   <leftBranch netRelationRef="nr_ne15ne16_swi84" branchingSpeed="0" joiningSpeed="0"
7     ↪  radius="-500"/>
8   <rightBranch netRelationRef="nr_ne14ne16_swi84" branchingSpeed="0" joiningSpeed="0"
9     ↪  radius="0"/>
10  </switchIS>
```

La clase *spotElementProjection* define la ubicación en el espacio del elemento ferroviario referido. En este caso, como se puede ver en el Código 2.16, la posición del cambio de vías simple es la coordenada (-561 ; -450).

Código 2.16: Clase *spotElementProjection* (*ordinarySwitch*)

```

1 <spotElementProjection refersToElement="swi84" id="vis01_sep16">
2   <name name="Sw01" language="en"/>
3   <coordinate x="-560.994" y="-450.000"/>
4 </spotElementProjection>
```

La clase *switchIL* (Código 2.17), definida dentro del vector de clases *switchesIL*, se encuentra dentro de la clase *assetsForIL*, en la clase *interlocking*. Contiene datos extra sobre el comportamiento dinámico del cambio de vías y define explícitamente los otros dos nodos, en contraposición a *switchIS* del cual hay que obtenerlos procesando un string. El RNA puede obtener los *netE-*

ment de ambas clase y compararlos, anulando el análisis si los *netElement* definidos en *switchIS* y *switchIL* no son coincidentes.

Código 2.17: Clase *SwitchIL* (ordinarySwitch)

```

1 <switchIL id="il_swi84" maxThrowTime="PT10S" typicalThrowTime="PT6S" isKeyLocked="false"
2   ↪ returnsToPreferredPosition="false">
3     <refersTo ref="swi84"/>
4     <branchLeft ref="ne15"/>
5     <branchRight ref="ne14"/>
6   </switchIL>
```

Los cambios de vías dobles (Código 2.18), a diferencia de los cambios de vías simples, siempre tienen dos ramas derechas (*straightBranch*) y dos ramas de bifurcación (*turningBranch*) definidas. Además, dentro de la definición de cada rama tenemos el atributo *netRelationRef*, del cual se puede obtener, procesamiento mediante, los otros *netElement* correspondientes a las ramas: ne41, ne82, ne84 y ne85. En este caso las posiciones del cambio permitirían circular entre alguna de las cuatro opciones: ne82 y ne84, ne41 y ne85, ne41 y ne82, ne84 y ne85.

Código 2.18: Clase *switchIS* (doubleSwitchCrossing)

```

1 <switchIS type="doubleSwitchCrossing" id="dsw328">
2   <straightBranch netRelationRef="nr_ne82ne84_dsw328" radius="0"/>
3   <straightBranch netRelationRef="nr_ne41ne85_dsw328" radius="0"/>
4   <turningBranch branchingSpeed="60" joiningSpeed="60" netRelationRef="
5     ↪ nr_ne41ne82_dsw328" radius="-500"/>
6   <turningBranch branchingSpeed="60" joiningSpeed="60" netRelationRef="
7     ↪ nr_ne84ne85_dsw328" radius="-500"/>
8   <designator register="_AdvancedExample" entry="SLIP_SWITCH_Sw05"/>
9   <spotLocation netElementRef="ne82" intrinsicCoord="1.0000" applicationDirection="
  ↪ normal" id="dsw328_sloc01"/>
  <name name="Sw05" language="en"/>
</switchIS>
```

La clase *spotElementProjection* define la ubicación en el espacio del elemento ferroviario referido. En este caso, como se puede ver en el Código 2.19, la posición del cambio de vías dobles es la coordenada (-300 ; 1200).

Código 2.19: Clase *spotElementProjection* (doubleSwitchCrossing)

```

1 <spotElementProjection refersToElement="dsw328" id="vis01_sep106">
2   <coordinate x="-300.129" y="1200.000"/>
3   <name name="Sw05" language="en"/>
4 </spotElementProjection>
```

Al igual que con los cambios de vías simples, el análisis de los cambios de vías dobles es complementando con el análisis de la clase *switchIL* (Código 2.20). El RNA puede obtener los *netElement* definidos en las clases *switchIS* y *switchIL*, compararlos y anular el análisis si los *netElement* no son coincidentes.

Código 2.20: Clase *SwitchIL* (doubleSwitchCrossing)

```

1 <switchIL maxThrowTime="PT10S" typicalThrowTime="PT6S" returnsToPreferredPosition="false"
```

```

1   ↳ isKeyLocked="false" id="il_dsw328_ne82_ne85">
2     <branchLeft ref="ne82"/>
3     <branchRight ref="ne85"/>
4     <refersTo ref="dsw328"/>
5     <relatedMovableElement ref="il_dsw328_ne84_ne41"/>
6   </switchIL>

```

Los cambios de vías en tijeras (Código 2.21), a diferencia de los cambios de vías simples o dobles, no posee ramas principales. Ambas ramas, definidas en *external*, pueden considerarse igualmente importantes. Además, dentro de la definición de cada rama es el atributo *Ref* y no *netRelationRef*, el que tiene la información de los *netElement*. En este caso: ne21 y ne104 pertenecen a una rama, mientras que ne52 y ne77 pertenecen a otra. En el caso de los cambios de vías en tijera, la ubicación del elemento se determina en base a la ubicación del *netElement* definido en *spotLocation* ya que su ubicación coincide al tener el parámetro *intrinsicCoord* igualado a uno.

Código 2.21: Clase *Crossing*

```

1 <crossing id="scr314">
2   <designator register="_AdvancedExample" entry="CROSSING_Sw03"/>
3   <external id="scr314_1" ref="nr_ne21ne104_scr314"/>
4   <external id="scr314_2" ref="nr_ne52ne77_scr314"/>
5   <spotLocation netElementRef="ne21" intrinsicCoord="1.0000" applicationDirection
       ↳ ="normal" id="scr314_sloc01"/>
6   <name name="Sw03" language="en"/>
7 </crossing>

```

La clase *movableCrossing* (Código 2.22), definida dentro del vector de clases *movableCrossings*, se encuentra dentro de la clase *assetsForIL*, en la clase *interlocking*. Contiene datos extra sobre el comportamiento dinámico del cambio de vías y define explícitamente los otros dos nodos, en contraposición a *crossings* del cual hay que obtenerlos procesando un string. El RNA puede obtener los *netElement* de ambas clase y compararlos, anulando el análisis si los *netElement* definidos en *crossings* y *movableCrossings* no son coincidentes.

Código 2.22: Clase *movableCrossing*

```

1 <movableCrossing preferredPosition="upleft-rightdown" maxThrowTime="PT1OS"
       ↳ returnsToPreferredPosition="false" isKeyLocked="false" id="il_scr314">
2   <branchUpLeft ref="ne21"/>
3   <branchUpRight ref="ne52"/>
4   <branchDownLeft ref="ne77"/>
5   <branchDownRight ref="ne104"/>
6   <refersTo ref="scr314"/>
7 </movableCrossing>

```

El RNA utiliza el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para detectar todos estos parámetros y crear un vector de máquinas de cambios (switches) indexado por el id de cada máquina de cambios (sw_id). La existencia y ubicación de las máquinas de cambios ya se habían obtenido mediante el análisis de la red de grafos ferroviaria. Los Algoritmos 3 y 4 analizan la clase *switchIS* y el Algoritmo 5 analizan la clase *crossings* para confirmar la existencia del cambio de vías. Luego se analiza la clase *spotElementProjection* para confirmar la ubicación del cambio de vías. Los datos obtenidos en *LeftBranch*, *RightBranch*, *straightBranch* y/o *turningBranch* permiten obtener los nodos de las ramificaciones que luego se conformarán analizando las clases *switchIL* o *movableCrossing*, según

corresponda, en algoritmos posteriores.

Algoritmo 3: Algoritmo detector de cambios de vías simples.

```

{switches} ← {}
if infrastructure.SwitchesIS != None then
    for i in infrastructure.SwitchesIS[0].SwitchIS do
        if i.Id not in switchesIS.keys() then
            sw_id ← i.Name[0].Name
            j ← i.SpotLocation[0]
            node ← j.NetElementRef
            type ← i.Type
            if type == 'OrdinarySwitch' then
                left_id ← i.LeftBranch[0].NetRelationRef
                right_id ← i.RightBranch[0].NetRelationRef
                switches[sw_id] ← {"Node":node}
                switches[sw_id] ← {"Continue":i.ContinueCourse}
                switches[sw_id] ← {"Branch":i.BranchCourse}
                switches[sw_id] ← {"Dir":j.ApplicationDirection}
                switches[sw_id] ← {"LeftBranch":j.left_id}
                switches[sw_id] ← {"RightBranch":j.right_id}
            end if
        end if
    end for
end if
visual_data ← visualization.Visualization
if visual_data != None then
    for i in visual_data[0].SpotElementProjection do
        sw_id ← i.Name[0].Name
        if 'Sw' in sw_id then
            pos_x ← int(i.Coordinate[0].X)
            pos_y ← int(i.Coordinate[0].Y)
            switches[sw_id] ← {"Position": [pos_x,-pos_y]}
        end if
    end for
end if
Return {switches}

```

Algoritmo 4: Algoritmo detector de cambios de vías dobles.

```

{switches} ← {}
if infrastructure.SwitchesIS != None then
  for i in infrastructure.SwitchesIS[0].SwitchIS do
    if i.Id not in switchesIS.keys() then
      sw_id ← i.Name[0].Name
      j ← i.SpotLocation[0]
      node ← j.NetElementRef
      type ← i.Type
      if type == 'DoubleSwitchCrossing' then
        straightBranch_A_id ← i.StraightBranch[0].NetRelationRef
        straightBranch_B_id ← i.StraightBranch[1].NetRelationRef
        turningBranch_A_id ← i.TurningBranch[0].NetRelationRef
        turningBranch_B_id ← i.TurningBranch[1].NetRelationRef
        for X,Z combination(A,B) do
          switches[sw_id+'XZ'] ← {"Node":node}
          switches[sw_id+'XZ'] ← {"Continue":i.ContinueCourse}
          switches[sw_id+'XZ'] ← {"Branch":i.BranchCourse}
          switches[sw_id+'XZ'] ← {"Dir":j.ApplicationDirection}
          switches[sw_id+'XZ'] ← {"LeftBranch":j.straightBranch_X}
          switches[sw_id+'XZ'] ← {"RightBranch":j.turningBranch_Z}
        end for
      end if
    end if
  end for
end if
visual_data ← visualization.Visualization
if visual_data != None then
  for i in visual_data[0].SpotElementProjection do
    sw_id ← i.Name[0].Name
    if 'dsw' in sw_id then
      pos_x ← int(i.Coordinate[0].X)
      pos_y ← int(i.Coordinate[0].Y)
      switches[sw_id] ← {"Position": [pos_x,-pos_y]}
    end if
  end for
end if
Return {switches}

```

Algoritmo 5: Algoritmo detector de cambios de vías en tijera.

```

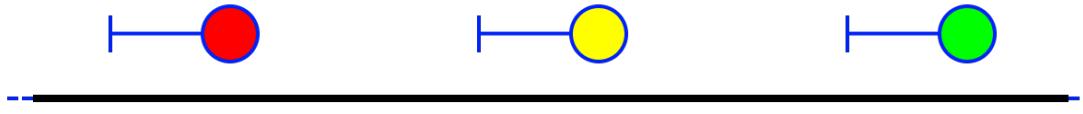
{switches} ← {}
if infrastructure.Crossings != None then
    for i in infrastructure.Crossings[0].SwitchIS do
        if i.Id not in Crossings.keys() then
            sw_id ← i.Name[0].Name
            j ← i.SpotLocation[0]
            node ← j.NetElementRef
            type ← i.Type
            direction ← i.SpotLocation[0].ApplicationDirection
            if type == 'DoubleSwitchCrossing' then
                straightBranch_A_id ← i.External[0].Ref
                straightBranch_B_id ← i.External[1].Ref
                turningBranch_A_id ← i.External[0].Ref
                turningBranch_B_id ← i.External[1].Ref
                for X,Z combination(A,B) do
                    switches[sw_id+'XZ'] ← {"Node":node}
                    switches[sw_id+'XZ'] ← {"Continue":i.ContinueCourse}
                    switches[sw_id+'XZ'] ← {"Branch":i.BranchCourse}
                    switches[sw_id+'XZ'] ← {"Dir":j.ApplicationDirection}
                    switches[sw_id+'XZ'] ← {"LeftBranch":j.straightBranch_X}
                    switches[sw_id+'XZ'] ← {"RightBranch":j.turningBranch_Z}
                end for
            end if
        end if
    end for
end if
visual_data ← visualization.Visualization
if visual_data != None then
    for i in visual_data[0].SpotElementProjection do
        sw_id ← i.Name[0].Name
        if 'scr' in sw_id then
            pos_x ← int(i.Coordinate[0].X)
            pos_y ← int(i.Coordinate[0].Y)
            switches[sw_id] ← {"Position": [pos_x,-pos_y]}
        end if
    end for
end if
Return {switches}

```

2.1.10. Señales ferroviarias

El sistema de señalamiento utiliza las señales ferroviarias para indicarle al conductor del tren si tiene autoridad de tránsito en el próximo tramo de vías y a qué velocidad se le permite circular; esto, por medio del color de las señales, 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, las señales ferroviarios cambian su aspecto en función de los eventos de los tramos siguientes. En la Figura 2.14 se presentan los posibles aspectos que pueden adoptar los semáforos utilizados en la gran mayoría de las líneas ferroviarias.

Peligro: obligación de **Precaución:** próxima señal puede **Vía libre:** continuar la detener la formación. ser roja, velocidad reducida. marcha.



Semáforos de tres aspectos

Peligro: obligación de detener la formación. **Precaución:** velocidad reducida.

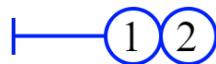


Semáforos de dos aspectos

1 semáforo



2 semáforos



3 semáforos



Múltiples semáforos en un solo poste

Figura 2.14: Aspectos de cada semáforo y múltiples señales en un semáforo.

El tipo más común de señal presenta tres aspectos, utilizado mayoritariamente en señales de circulación en vías principales. También existen las señales de maniobras, que se utilizan en cambios de vías donde, por su peligrosidad, solo se podrían permitir aspectos rojos (prohibido avanzar) y amarillos (precaución). Algunas líneas, como la Línea Roca, utilizan señales de cuatro aspectos, que poseen un doble amarillo antes del amarillo simple, para permitir así tramos de vías más cortos de forma segura.

Varias señales pueden compartir físicamente un mismo poste, constituyendo un semáforo, como se visualiza en la Figura 2.14. De ahora en adelante, siempre que hablemos de un semáforo nos referimos al objeto físico o a un conjunto de señales que comparten un poste, mientras que hablaremos de señal como concepto abstracto o para referirnos a un solo semáforo del poste.

A la hora de leer las señales de los semáforos de la Figura 2.14, consideramos al primer semáforo como el correspondiente a la rama principal de la red y a continuación todas la ramificaciones de derecha a izquierda, en el sentido de circulación de la formación. Esto permite simplificar el diagrama de señalamiento, concentrando varias señales que comparten una misma posición en un solo símbolo. Físicamente, las señales que comparten posiciones también comparten un poste o semáforo.

En la Figura 2.15 se ilustra la correspondencia de cada señal en el poste indicado como S01. Los colores del diagrama son meramente para indicar la correspondencia y no representan ningún aspecto. La señal S01 A (rosa) permite la circulación por la vía principal hasta la señal S02. La señal

S01 B (verde) permite la circulación por la primer ramificación, hasta la señal S03. Finalmente, la señal S01 C (roja) permite la circulación por la vía secundaria mas a la izquierda de la formación, hasta la señal S04.

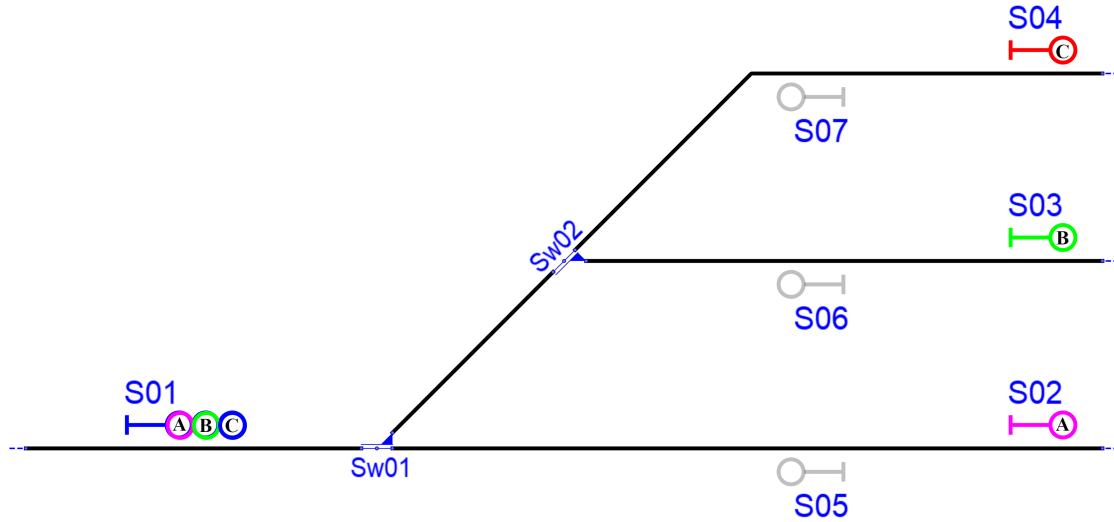


Figura 2.15: Ejemplo de asignación de señalización ramificada.

Los semáforos siempre deben situarse a la izquierda del sentido de circulación. Por ejemplo, los semáforos S01, S02, S03 y S04 de la Figura 2.15 son señales para un tren circulando de izquierda a derecha. Por otro lado, los semáforos S05, S06 y S07 son señales para un tren circulando de derecha a izquierda.

2.2. Algoritmos de generación de señalamiento

En esta Sección se abordará la generación automática de señalamiento que realiza el RNA para cada uno de los elementos ferroviarios básicos y se mostrará como los principios de señalamiento (ver Sección 1.2) se cumplen para cada uno de ellos de forma individual. Se asumirá para el análisis de la red ferroviaria que todas las vías son bidireccionales y que los elementos ferroviarios deben ser protegidos en todas las direcciones. De esta forma, siempre se tendrá un análisis completo de la red ferroviaria, contemplando todos los posibles usos de la red que el operador pueda darle.

Tal como se explico en la Sección 1.7.3, al ser el modelo ferroviario de grafos un sistema lineal, podemos definir un principio de superposición de señales. Las señales generadas para un sistema con elementos ferroviarios A y B serán la superposición de las señales generadas para proteger al elemento A y las señales generadas para proteger al elemento B. No obstante, puede darse el caso de que una señal que originalmente protegía al elemento A también cumpla la función de proteger a B, por lo cual existe un solapamiento de señales. Por lo tanto, el señalamiento final tendrá menos señales que la suma de los señalamientos originales. Podemos generalizar esta conclusión en la Fórmula 2.1 y Fórmula 2.2:

$$\text{señales}(a,b,\dots,z) = \sum_{\text{elemento}=a}^z \text{señales}(\text{elemento}) - \bigcap_{\text{elemento}=a}^z \text{señales}(\text{elemento},\text{elemento}+1) \quad (2.1)$$

$$\text{señales}(a,b,\dots,z) \leq \sum_{\text{elemento}=a}^z \text{señales}(\text{elemento}) \quad (2.2)$$

Es decir, el señalamiento total siempre será menor a la suma del señalamiento generado para cada uno de los elementos ferroviarios, salvo que no existan señales solapadas. Para obtener el señalamiento final, el RNA debe no solo generar el señalamiento para cada elemento y combinarlo, sino también detectar solapamientos, inconsistencias o señales contradictorias que puedan comprometer la seguridad de la red y eliminarlas. La eliminación de señales deberá realizarse con ciertos criterios, para no disminuir el nivel de seguridad de la red ferroviaria. El proceso de simplificación del señalamiento se abordará en detalle en la Sección H.3.

2.2.1. Señalización en fin de vía y transiciones

Tal como se definió en la Sección 2.1.5, las redes ferroviarias presentan tanto fines de vía absolutos (modelados en railML por la clase *bufferStop*) como fines de vía relativos (modelados en railML por la clase *lineBorder*). El RNA detecta estos elementos y generará el señalamiento correspondiente en base a los principios de señalamiento expuestos en la Sección 1.2.

En el caso de las transiciones para fines de vía relativos, por el principio de infraestructura (P_6), es necesario que existan señales que indiquen que la formación pueda circular por la transición de una región ferroviaria a otra. Esta señal debe otorgar autoridad a la formación de forma unívoca, por el principio de claridad (P_2). Adicionalmente, esa señal debe situarse con suficiente antelación a la transición, por el principio de anticipación (P_3). Aplicando el principio de no bloqueo (P_7) regulamos la circulación entre dos regiones ferroviarias para que puedan circular sin demoras ni atascos. Finalmente, el principio de granularidad (P_4) y autoridad (P_1) nos pide que el derecho de uso de la infraestructura debe ser acotado, pero funcional. Es decir, la autoridad debe aplicarse en una sección con un mínimo de largo como para que tenga sentido.

En el Algoritmo 6 aplicamos todos estos principios al solicitar que la sección donde colocamos la señal tenga un parámetro de largo mínimo. Además, al ser una transición, las señales colocadas son todas de partida, para transitar de una región a la otra, de ser permitido, o detener la formación en la frontera entre ambas regiones, de encontrarse saturada la próxima región.

Algoritmo 6: Algoritmo de generación de señalamiento para Line borders.

```

1 for netElement WITH LineBorder do
2   if netElement.Length > FIXED_LENGTH then
3     if NOT EXIST next netElement then
4       [Signals] ← ADD departure signal >>>
5     if NOT EXIST prev netElement then
6       [Signals] ← ADD departure signal <<<
```

Result: [Signals]

En el caso de los *buffer stops*, por el principio de terminalidad (P_5) es necesaria una señal que indique a las formaciones que deben detenerse antes de llegar al final de la vía. Además, es necesaria una señal de partida en sentido contrario para que las formaciones reanuden su marcha en el otro sentido. Esto fue implementado en el Algoritmo 7. La restricción de tamaño fue removida porque el final de vía y la formación deben ser protegidos sin importar el largo de la sección.

Algoritmo 7: Algoritmo de generación de señalamiento para Buffer stops.

```

1 for netElement WITH BufferStops do
2   if NOT EXIST next netElement then
3     [Signals] ← ADD stop signal >>>
4     [Signals] ← ADD departure signal <<<
5   if NOT EXIST prev netElement then
6     [Signals] ← ADD stop signal <<<
7     [Signals] ← ADD departure signal >>>

```

Result: [Signals]

Aplicando el Algoritmo 6 y el Algoritmo 7 a dos vías paralelas con finales de vías relativos (vía superior) y absolutos (vía inferior) obtenemos un señalamiento como el que se visualiza en la Figura 2.16.

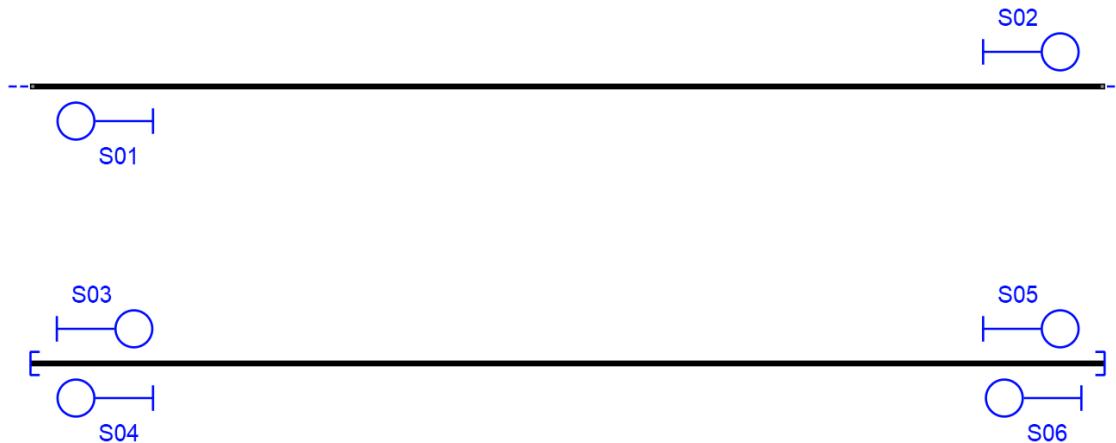


Figura 2.16: Señalamiento generado para finales de vía relativos y absolutos.

El RNA detecta los *line borders* en la vía superior y aplica el Algoritmo 6, generando las señales de partida S01 para las formaciones que se mueven de derecha a izquierda y la señal de partida S02 para las formaciones que se mueven de izquierda a derecha, saliendo de la región mostrada en la Figura 2.16. Además, al detectar los *buffer stops* en la vía inferior, el RNA aplica el Algoritmo 7, generando cuatro nuevas señales. Las señales S04 y S05 son señales de parada para indicar a las formaciones que deben detenerse antes de colisionar con el final de la vía al transitar de derecha a izquierda o viceversa, respectivamente. Las señales S03 y S06 permiten a las formaciones reanudar su marcha en sentido contrario al que venían circulando, retomando su lugar en la red ferroviaria.

2.2.2. Detectores de posición y velocidad.

En la Sección 2.1.6 definimos dos elementos ferroviarios utilizados para detectar la posición de una formación: los contadores de ejes y los circuitos de vía. Adicionalmente, los contadores de ejes se utilizan para calcular la velocidad de la formación. Es claro que, para no alterar la medición de velocidad, no tiene sentido detener la marcha de la formación próximo a un contador de ejes y por lo tanto el principio de anticipación (P_3) no se aplica. Adicionalmente, los contadores de ejes no representan un peligro para la formación, al encontrarse en la cara interna de la vía no es posible para una formación colisionar con ellos, por lo que cubrimos el principio de infraestructura (P_6)

sin utilizar ninguna señal. En conclusión, el RNA no requiere agregar ninguna señal asociada a los contadores de ejes.

Utilizamos a las junturas como los elementos divisorios entre los circuitos de vía a los cuales aplicaremos el principio de antelación y autoridad (P_1). Por el principio de antelación, las señales deben colocarse antes de cada juntura para indicar con suficiente tiempo si se puede proseguir la marcha o no hacia una nueva sección. Además, como solo una formación a la vez puede ocupar cada circuito de vía, el principio de autoridad (P_1) se aplica para restringir el acceso a las próximas secciones de encontrarse ocupadas. Finalmente, por el principio de no bloqueo (P_7), podemos asumir que no tiene sentido detener la marcha de las formaciones en secciones muy pequeñas, incluso mas pequeñas que una formación, por lo que solo se aplicaran señales en secciones mayores a un tamaño determinado.

En el Algoritmo 8 se define la asignación de señalamientos para los contadores de ejes y circuitos de vía limitados por las junturas. El parámetro de largo mínimo para la sección puede ser modificado por el operador, pero es único para toda la red. De esta forma nos aseguramos, además, que redes ferroviarias de gran tamaño pero sin demasiados elementos ferroviarios pueda tener señalamiento cada cierta cantidad de metros o kilómetros. Ya que toda red ferroviaria esta constituida por rieles que presentan junturas cada ciertas distancias para evitar que la expansión térmica provoque descarrilamientos.

Algoritmo 8: Algoritmo de generación de señalamiento para *Axle Counters y Rail Joints*

```

1 if netElement WITH AxleCounters then
2   // Do nothing
3 for netElement WITH RailJoints do
4   Track.Length = Length ( between RailJoints )
5   if Track.Length > FIXED_LENGTH then
6     [Signals] ← ADD circulation signal >>>
7     [Signals] ← ADD circulation signal <<<
```

Result: [Signals]

Aplicando el Algoritmo 8 a un sistema de tres vías paralelas se obtiene un resultado como el ilustrado en la Figura 2.17. La vía superior presenta un contador de ejes, la vía intermedia dos circuitos de vía, uno corto y uno largo. La vía inferior presenta dos circuitos de vía, ambos de larga longitud.



Figura 2.17: Señalamiento generado para contadores de ejes y circuitos de vías.

La vía superior incluye al contador de ejes 05 y no se le asigna ninguna señal. La vía intermedia tiene dos secciones, la primera es la correspondiente al circuito de vía 01 y no se le asigna una señal al ser muy corta, la segunda es la correspondiente al circuito de vía 02 y se le asigna la señal S03 antes de la juntura J01. La señal S03 se utiliza para detener la formación que transita de derecha a izquierda antes de ingresar a la sección del circuito de vía 01 y reiniciar la marcha si esta última se encuentra desocupada. Finalmente, las señales S01 y S02 se asignan antes de la juntura J02 para poder transitar desde la sección del circuito de vía 03 a la sección del circuito de vía 04 y viceversa.

2.2.3. Plataformas ferroviarias

En la Sección 2.1.7 definimos la clase *platform* que modela a las plataformas ferroviarias. Las plataformas ferroviarias son un punto del recorrido donde las formaciones pueden detenerse, algunas veces de forma opcional dependiendo el itinerario, para que los pasajeros desciendan y nuevos pasajeros puedan ascender. Claramente existe una limitación de autoridad: las formaciones necesitan un nuevo permiso para continuar circulando una vez alcanzada la plataforma. Permiso que será otorgado o negado según el estado del sistema a continuación del recorrido. Las plataformas ferroviarias suelen encontrarse en zonas pobladas, cerca de otras infraestructuras, zonas comerciales o residenciales, por lo que avisar con antelación al maquinista que debe disminuir la marcha y/o detenerse es esencial.

Además, es posible que las formaciones convivan con otras formaciones que también hacen uso de la estación, por lo que las señales deben ser únicas y claras. Algunas estaciones pueden contener fines de vías o ramificaciones hacia talleres u otros ramales, por lo que también se aplica el principio de terminalidad (P_5) y granularidad (P_4). Finalmente, es importante el mantener una circulación fluida de las formaciones, de modo de no retrasar el itinerario de las formaciones que vienen detrás, por lo que se deberá aplicar el principio de no bloqueo (P_7).

En el Algoritmo 9 definimos a la señal de partida como la única señal necesaria para operar una plataforma, asumiendo que la señal de ingreso de la misma será dada por otra instancia previa. De no existir otro elemento cercano por el cual se genere una señal, se puede asumir que la distancia a la plataforma es muy larga y, por lo tanto, se aplicaría el Algoritmo 8, protegiendo a la plataforma.

En el caso de que la vía sea bidireccional se añadirán señales de partida para ambos sentidos.

Algoritmo 9: Algoritmo de generación de señalamiento para *platforms*.

```

1 for netElement WITH Platform do
    /* Before leaving platform from left */
    [Signals] ← ADD departure signal >>>
    /* After leaving platform from right */
    [Signals] ← ADD departure signal <<<

```

Result: [Signals]

Aplicando el Algoritmo 9 a un sistema de dos vías paralelas con plataformas pertenecientes a la misma estación obtenemos el resultado ilustrado en la Figura 2.18. Se asumieron que ambas vías son bidireccionales, en caso contrario solo se generarían las señales S01 y S02.

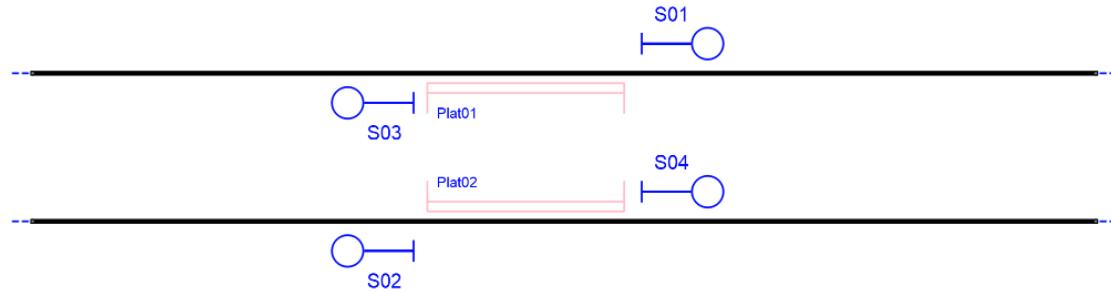


Figura 2.18: Señalamiento generado para estaciones ferroviarias.

Una formación que circule de izquierda a derecha deberá detenerse antes de la señal S01, en caso de utilizar la vía superior, o antes de la señal S04, en caso de utilizar la vía inferior. Análogamente, las formaciones deberán detenerse antes de las señales S03 y S02 en el caso de transitar de derecha a izquierda por la vía superior o inferior respectivamente. Sólo cuando estas señales otorguen a la formación autoridad para circular podrán reanudar su marcha hasta la próxima señal disponible, fuera del alcance de lo ilustrado en la Figura 2.18.

2.2.4. Cruces ferroviarios

En la Sección 2.1.8 definimos la clase *levelCrossingIs* que modela a los cruces ferroviarios a nivel. Los cruces ferroviarios son un sector crítico de los sistemas ferroviarios ya que conviven tanto las formaciones como los vehículos y peatones. El cruce ferroviario impone un límite a la autoridad ya que es necesario confirmar que la barrera ferroviaria se encuentre baja antes de habilitar a una formación. Además, podemos utilizar al cruce ferroviario como frontera entre dos rutas, aplicando el principio de granularidad (P_4). Esto se refuerza con el principio de infraestructura (P_6) ya que, aún cuando la formación no puede dañar la barrera ferroviaria al transitar el paso a nivel sin autoridad, puede dañar severamente a los conductores de vehículos o peatones.

Por el principio de anticipación (P_3), las señales se colocan a varios metros de distancia del cruce ferroviario. Las barreras ferroviarias comienzan su descenso al detectar la presencia de una formación bastantes metros antes de estas señales. De esta manera la formación llegará al cruce ferroviario cuando las barreras ya hayan descendido. De igual forma, las barreras ferroviarias suelen complementarse con indicaciones lumínicas y sonoras que se activan al comienzo del descenso. Para asegurar una circulación fluida, por el principio de granularidad (P_4), las formaciones deberán tener una mayor prioridad que los vehículos y peatones, haciendo descender la barrera siempre que una formación se aproxime, sin excepciones.

En el Algoritmo 10 definimos a la señal de circulación como la única señal necesaria para operar un cruce ferroviario, protegiendo a todos los vehículos, peatones y formaciones que quieran hacer uso de la intersección. En el caso de que la vía sea bidireccional se añadirán señales de circulación para proteger el cruce en ambos sentidos.

Algoritmo 10: Algoritmo de generación de señalamiento para *levelCrossingIs*.

```

1 for netElement WITH LevelCrossing do
    /* Before reaching level crossing */
    [Signals] ← ADD circulation signal >>>
    /* After leaving level crossing */
    [Signals] ← ADD circulation signal <<<

```

Result: [Signals]

Aplicando el Algoritmo 10 a un sistema de dos vías paralelas con un cruce de ferroviario que atravesie perpendicularmente ambas vías obtenemos el resultado ilustrado en la Figura 2.19. Se asumieron que ambas vías son bidireccionales, en caso contrario solo se generarían las señales S01 y S02.



Figura 2.19: Señalamiento generado para cruces ferroviarios.

Una formación que circule de izquierda a derecha deberá detenerse antes de la señal S01, en caso de utilizar la vía superior, o antes de la señal S04, en caso de utilizar la vía inferior. Las señales presentarán un aspecto rojo si las barreras se encuentren altas o en proceso de descender. Análogamente, las formaciones deberán detenerse antes de las señales S03 y S02 en el caso de transitar de derecha a izquierda por la vía superior o inferior respectivamente. Cuando las barreras se encuentren bajas, los aspectos de las señales cambiarán a verde para habilitar la circulación de la formación por sobre el paso a nivel, continuando su marcha hasta la próxima señal disponible, fuera del alcance de lo ilustrado en la Figura 2.19.

2.2.5. Cambios de vías

En la Sección 2.1.9 definimos la clase *switchIs* que modela a los cambios de vías. Los cambios de vías son de los elementos ferroviarios más críticos de la red. Por lo tanto, se deberá aplicar al principio de infraestructura (P_6) a cada una de sus entradas. Además, las máquinas de cambios actúan como una frontera entre las ramas principales de la red, donde se circula a mayor velocidad, y las ramas secundarias, donde se circula a una velocidad menor. Esto divide las rutas en dos o más rutas, aplicando el principio de autoridad (P_1) y granularidad (P_4). Finalmente, por el principio de no bloqueo (P_7), será necesario generar señales de diferente tipo para las ramas principales y las secundarias, priorizando las primeras. Esto puede cumplirse si otorgamos señales de circulación de tres o más aspectos a la rama principal y señales de maniobra para utilizar una rama secundaria como salida o entrada de la red.

En el Algoritmo 11 definimos al punto de acceso de inicio como la vía desde la cual la trayectoria de la formación se bifurca, para lo cual se genera una señal de circulación para continuar por la rama principal y una señal de maniobra para acceder a la rama secundaria. Además, se añade una señal de circulación en la rama principal y una señal de maniobra en la rama secundaria, ambas en sentido contrario, para poder retornar al punto de inicio.

Algoritmo 11: Algoritmo de generación de señalamiento para *Switches*

```

1 for Switch in Switches do
    /* All signals must point to switch */
    switch Switch.Type do
        case Start do
            [Signals] ← ADD circulation signal
            [Signals] ← ADD maneuver signal
        case Continue branch do
            [Signals] ← ADD circulation signal
        case Detour branch do
            [Signals] ← ADD maneuver signal

```

Result: [Signals]

Aplicando el Algoritmo 11 a un sistema de vías que se bifurcan obtenemos el resultado ilustrado en la Figura 2.20. Se asumieron que ambas vías son bidireccionales. En caso contrario, si el sentido de circulación es únicamente de bifurcación, las señales de retorno S02 y S03 no son necesarias. Si el sentido de circulación fuese de derecha a izquierda, las señales de inicio S01 no son necesarias.

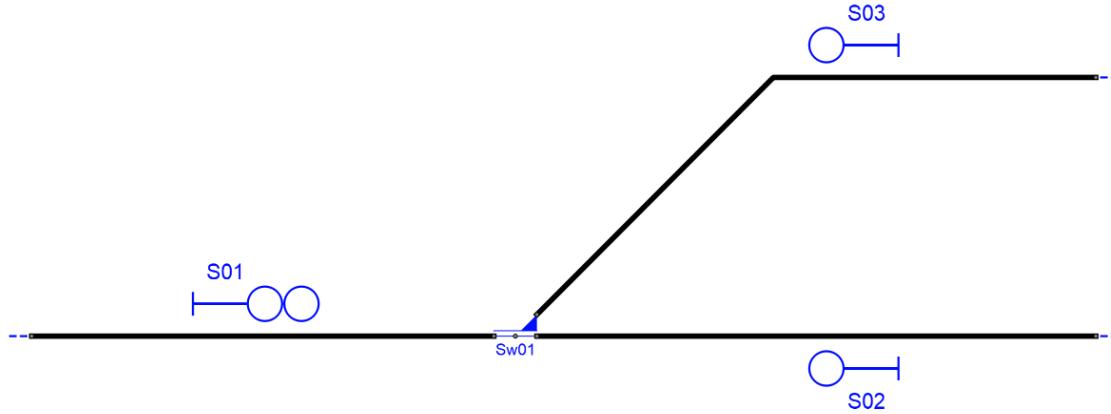


Figura 2.20: Señalamiento generado para cambios de vías.

Una formación que circule de izquierda a derecha deberá detenerse antes de la señal S01. El aspecto mas cercano al poste de la señal S01 indicará si la formación puede circular por la vía principal, como se explicó en la Sección 2.1.10, previa confirmación por parte del sistema de enclavamiento de que el cambio Sw01 se encuentra en posición normal. De la misma forma, el sistema de enclavamiento confirmará que el cambio Sw01 se encuentra en posición reverso previo a habilitar la circulación por la vía secundaria mediante el aspecto mas alejado al poste de la señal S01.

Una formación circulando de derecha a izquierda hará uso de las señales S02 y S03 si se encuentran en las vías principal o secundaria respectivamente. Las señales se habilitarán si el sistema de enclavamientos confirma que el cambio Sw01 se encuentra en la posición normal o reverso res-

pectivamente y si no hay formaciones ocupando la vía de inicio donde se encuentra la señal S01, continuando su marcha hasta la próxima señal disponible, fuera del alcance de lo ilustrado en la Figura 2.20.

2.3. Algoritmos de simplificación de señalamiento

Una vez finalizada la generación de señalamiento para cada elemento ferroviario, es necesario abordar el segundo miembro de la Fórmula 2.1 explicada al comienzo de la Sección 2.2. El señalamiento total para un sistema de N elementos ferroviarios será la suma de las señales necesarias para proteger a esos N elementos ferroviarios. Es inevitable que algunas señales se solapen con otras espacial o funcionalmente. Incluso algunas señales podrían contradecir a otras, oponiéndose al principio de claridad. La abundancia de señales puede hacer las rutas tan cortas que la autoridad estaría limitada a pequeñas porciones de la red, haciendo inviable una circulación fluida sobre la red.

La simplificación del señalamiento se sustenta en dos algoritmos básicos: simplificación por herencia vertical y simplificación por herencia horizontal. Los dos algoritmos pueden aplicarse en cualquier orden, aunque los algoritmos de herencia vertical solamente se pueden aplicar en redes ramificadas que tengan máquinas de cambios.

El proceso de simplificación finaliza con el algoritmo de limpieza, que elimina todas las señales redundantes en base a un nivel de prioridades. Estas prioridades se relacionan directamente con qué elemento ferroviario el RNA intentaba proteger con esta señal, reteniendo la señal más importante y eliminando el resto. También se combinan señales muy próximas en una vecindad y se borran señales contradictorias o redundantes, para obtener el señalamiento definitivo.

2.3.1. Algoritmo de simplificación por herencia vertical

Como se explicó en la Sección 2.1.9, un cambio de vías bifurca la vía principal en dos: una vía que continúa la rama principal y otra vía que se convierte en una rama secundaria. Tanto la vía de continuación como la vía ramificada pueden tener otra máquina de cambios que, a su vez, vuelve a dividir el trazado ferroviario en dos. Esto incrementa más y más el nivel de complejidad de la red, añadiendo caminos divergentes al principal que luego, mas adelante, podrían, o no, volver a converger utilizando mas máquinas de cambios. Cada divergencia aporta un nivel de profundidad a la red, mientras que cada convergencia reduce en uno el nivel de profundidad.

Un cambio de vías que tiene en cualquiera de sus ramas divergentes el nodo de inicio de otro cambio de vías a una distancia pequeña es un cambio compuesto. Los cambios compuestos requieren considerar a ambas máquinas de cambios como una sola, ya que es necesario posicionar ambos mecanismos para completar un movimiento. El ejemplo de La Figura 2.21 ayudará a clarificar el concepto de cambios compuestos e introducir el Algoritmo de simplificación por herencia vertical.

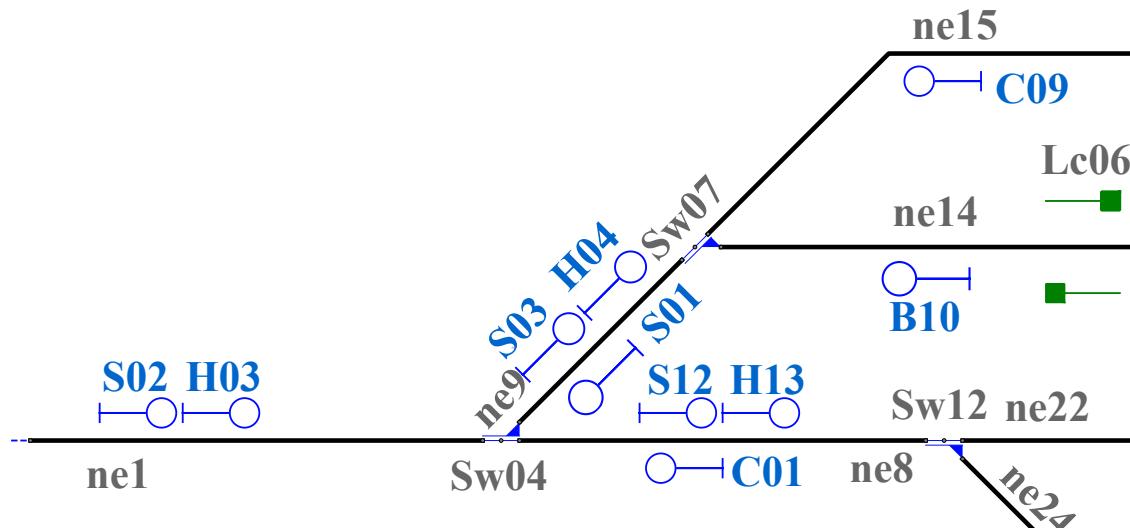


Figura 2.21: Señalamiento generado, simplificado por algoritmo de herencia vertical.

En la Sección H.3 se introdujo el Algoritmo 11 que define la asignación de señalamiento para los cambios de vías: una señal de circulación (S) y de maniobra (H) para el nodo de inicio (S), una señal de circulación para el nodo de continuación (C) y una señal de maniobras para el nodo de desvío (B). El Algoritmo 11 se aplicó de forma independiente a las máquinas de cambios Sw04, Sw07 y Sw12 y se obtuvo el señalamiento de la Figura 2.21. No obstante, el señalamiento generado no cumple con el principio de no bloqueo, al situar las señales S03, H04, B01, S12, H13 y C01 en las secciones correspondientes a los netElements ne9 y ne8. De esa manera, las formaciones podrían detenerse en esas secciones si las mencionadas señales presentasen un aspecto rojo.

Una formación detenida en la sección correspondiente al netElement ne9 no permitiría mover el cambio de vía Sw04, al tener parte de la formación aún en ne1. Tampoco permitiría accionar el cambio de vías Sw07 al no haber finalizado el movimiento. Análogamente, lo mismo ocurriría para una formación detenida en la sección correspondiente al netElement ne8. Es necesario que el movimiento finalice, despejando las secciones y permitiendo que otras formaciones utilicen los cambios de vías Sw04 y Sw07, o cualquier combinación de más de un cambio de vías.

Para solucionar el problema expuesto, el RNA aplica el Algoritmo 15 de simplificación por herencia vertical para reposicionar las señales situadas en zonas conflictivas. A diferencia del señalamiento explicado en la Sección 2.2, las señales que protegen a un cambio de vías B situado en una rama más profunda de la red no se encuentran inmediatamente precediendo al cambio de vías B. Estas señales son movidas a ramas de menor profundidad con suficiente espacio físico para contenerlas, situándose precediendo a otro cambio de vías A que decimos que ha *heredado* el señalamiento del cambio de vía B.

Algoritmo 12: Algoritmo de simplificación por herencia vertical

```

1 for each Signal protecting a Sw_A do
2   for each Sw_B != Sw_A do
    /* Criteria #0 */
    if distance(Sw_A,Sw_B) > MIN_DISTANCE then
      Continue
    if S in Sw_A.Branch also in Sw_B.Start then
      /* Criteria #1 */
      if S points to Sw_B.Start then
        Move S to Sw_A.Start
      /* Criteria #2 */
      if S points to Sw_A.Branch then
        Move S to Sw_B.Continue
        Move S to Sw_B.Branch
    if S in Sw_A.Continue also in Sw_B.Start then
      /* Criteria #3 */
      if S points to Sw_B.Start then
        Move S to Sw_A.Start
      /* Criteria #4 */
      if S points to Sw_A.Continue then
        Move S to Sw_B.Continue
        Move S to Sw_B.Branch
  
```

Result: [Signals]

Como puede apreciarse en el Algoritmo 15, existen cinco criterios a considerar para aplicar la herencia vertical. El primer criterio es excluyente, si la distancia entre ambas máquinas de cambios supera un parámetro determinado entonces el algoritmo no se aplica, al existir suficiente distancia para situar el señalamiento sin generar obstrucciones debido a formaciones detenidas entre ambos. Cuando la distancia entre las máquinas de cambios es menor a lo esperado, se aplican los otros cuatro criterios, en función de que nodos conectan entre sí la vía que comparten.

Definiendo al cambio de vías A como el cambio principal y el cambio de vías B como el cambio secundario situado en alguna de las ramificaciones del cambio de vías A, podemos diferenciar dos casos: que el señalamiento se encuentre en la rama de continuación del cambio de vías A o se encuentre en la rama de desvío. Además, este señalamiento puede estar apuntando al cambio de vías A o al cambio de vías B, protegiéndolo respectivamente.

El criterio para el desplazamiento del señalamiento busca cumplir el principio de Anticipación, el principio de Infraestructura y el principio de no bloqueo. Para eso, las señales son adelantadas, desplazándolas en sentido opuesto al que apuntan. Si el señalamiento se encuentra en la rama de desvío o continuación del cambio de vías A es indistinto, lo importante es a donde apuntan las señales. Si estas apuntan hacia el nodo de inicio del cambio de vías B, deben ser desplazadas, sin alterar su cantidad, hacia el nodo de inicio del cambio de vías A.

En el caso particular de que las señales en la zona conflictiva apunten hacia el cambio de vías A, tanto a su nodo de desvío como al nodo de continuidad, las mismas deberían moverse en sentido contrario al que apuntan. Sin embargo, en esa dirección la red se ramifica en dos debido al cambio de vías B. En este caso, las señales mueven duplicadas, a la rama de continuación y a la rama de desvío del cambio de vías B, en su totalidad.

Aplicando el Algoritmo 15 de herencia vertical al ejemplo de la Figura 2.21 obtenemos el señalamiento de la Figura 2.22. Las señales S03, H04, B01, S12, S13 y C01 serán adelantadas

para despejar las secciones correspondientes a los netElements ne9 y ne8. Debido a esto, los tres cambios de vías se usarán de forma complementaria, creando los siguientes cambios compuestos: $Sw_{04}^R + Sw_{07}^N$, $Sw_{04}^R + Sw_{07}^R$, $Sw_{04}^N + Sw_{12}^N$ y $Sw_{04}^N + Sw_{12}^R$. Los cuales se leen, considerando el ejemplo del cambio compuesto $Sw_{04}^R + Sw_{07}^N$, de la siguiente manera: el cambio de vías Sw_{04} en posición normal y el cambio de vías Sw_{07} en posición reversa.

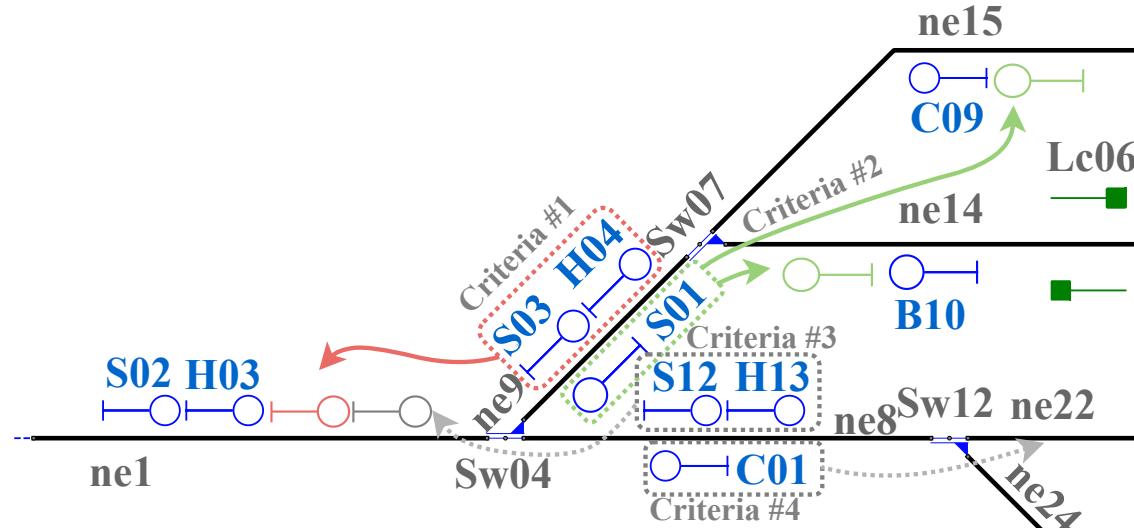


Figura 2.22: Señalamiento generado, simplificado por algoritmo de herencia vertical.

En el caso de las señales S03 y H04 que apuntan al nodo de inicio del cambio de vías Sw07, son desplazadas en sentido contrario al que apuntan, aplicando el criterio #1. Estas señales son movidas hasta el nodo de continuación del cambio de vías Sw04, correspondiente al netElement ne1, protegiendo el inicio del cambio compuesto $Sw_{04}^R + Sw_{07}^N$ y $Sw_{04}^R + Sw_{07}^R$. A estas señales se le suman las señales S12 y H13 situadas en la sección correspondiente al netElement ne8 que, aplicando el criterio #3, son desplazadas al nodo de inicio del cambio de vías Sw04, protegiendo el inicio del cambio compuesto $Sw_{04}^N + Sw_{12}^N$ y $Sw_{04}^N + Sw_{12}^R$.

La señal B01 apunta al nodo de desvío del cambio de vías Sw04 y, por el criterio #2, debe ser duplicada y movida. Una señal se posiciona en la sección correspondiente al netElement ne15 y la otra a la sección asociada al netElement ne14. De esta manera, la señal S01a y S01b seguirán protegiendo el nodo de desvío del cambio de vías Sw04, pero previamente tendrán que hacer uso del cambio de vías Sw07 en cualquiera de sus dos posiciones, protegiendo los nodos de cambio y continuación del cambio compuesto $Sw_{04}^R + Sw_{07}^N$ y $Sw_{04}^R + Sw_{07}^R$. De igual manera, la señal C01 es movida y duplicada a las secciones correspondientes a los netElements ne22 y ne24, debido al criterio #4. Estas nuevas señales C01a y C01b continúan protegiendo el nodo de continuación del cambio de vías Sw04, pero ahora deberán, adicionalmente, hacer uso del cambio de vías Sw12, protegiendo los nodos de continuación y desvío del cambio compuesto $Sw_{04}^N + Sw_{12}^N$ y $Sw_{04}^N + Sw_{12}^R$.

El Algoritmo 15 de herencia vertical es exitoso a la hora de despejar las zonas críticas entre cambios de vías muy cercanos. No obstante, también incrementa la cantidad de señales, como en el caso de las señales S01 y C01 al ser desplazadas en sentido divergente por los cambios de vías Sw07 y Sw12. Además, el Algoritmo 15 de herencia vertical no es aplicable solo a una dupla de cambios de vías, sino que puede ser aplicado para cualquier conjunto de cambios de vías. Esto provoca que las señales se hereden de forma iterativa hasta encontrar un cambio de vías con suficiente espacio para albergar las señales. Debido a esto, la cantidad de señales al comienzo de un cambio de vías previo a una ramificación profunda de la red tendrá una gran cantidad de señales.

En el caso del ejemplo de las Figuras 2.21 y 2.22, el algoritmo concluye con seis señales en el nodo de inicio de los cuatro cambios compuestos. Estas seis señales deberán habilitar cuatro caminos a transitar desde ne1: hacia ne15, hacia ne14, hacia ne22 y hacia ne24. Es claro que, teniendo mas señales que caminos a habilitar, será necesario un algoritmo de limpieza de señales duplicadas y/o que ya no tengan una utilidad en el señalamiento. El proceso de simplificación es explicado en profundidad en la Sección 2.3.3.

2.3.2. Algoritmo de simplificación por herencia horizontal

Todos los algoritmos de generación de señalamiento explicados en la Sección 2.2 colocan el señalamiento a una distancia fija, que llamaremos X, del elemento que buscan proteger. Existe, por lo tanto, una región espacial del tendido ferroviario anterior y posterior al elemento ferroviario donde se situarán las señales, que denominaremos región de señalamiento.

Si la distancia entre dos elementos ferroviarios es menor a $2X$ pero mayor a X , las señales generadas para proteger la región de señalamiento que comparten ocupará el mismo espacio físico. En este caso, el principio anticipación no se cumple porque las señales correspondientes se entremezclan, no dando suficiente tiempo para informar al conductor que un nuevo elemento ferroviario se encuentra delante. Incluso las señales podrían alterar su orden en esta región, avisando primero del próximo elemento antes de anunciar la partida del elemento que se está transitando en ese momento.

Si la distancia entre dos elementos ferroviarios es menor a X , las señales generadas para proteger la región de señalamiento que comparten ocupará el espacio físico correspondiente al otro elemento. Esto repercute negativamente en la claridad de las autoridades otorgadas a los conductores, impidiendo que se cumpla el principio de claridad. Tampoco se cumpliría el principio de anticipación y de no bloqueo, al no avisar con antelación la existencia de los peligros ni permitiría una circulación fluida al detener la marcha en el medio de otro elemento.

El RNA aplica el Algoritmo 15 de herencia horizontal para solucionar este inconveniente, analizando los objetos cercanos de a pares. Cuando la distancia entre dos objetos es menor a un parámetro dado, las señales son movidas y "heredadas" por los elementos anteriores o posteriores. Las señales de advertencia, previas a un elemento ferroviario, son anticipadas, salteando el elemento que provoca el solapamiento de señales. Las señales de partida son retrasadas, mas allá del elemento que provoca el solapamiento de señales. El resultado obtenido es que ambos elementos ferroviarios, al situarse tan próximo uno del otro, se consideran prácticamente como un solo elemento ferroviario a proteger.

Algoritmo 13: Algoritmo de simplificación por herencia horizontal

```
1 Given {Signals}, {objects}
2 for object_A in objects do
3   for object_B in objects do
4     if object_A != object_B then
5       if dist(object_A, object_B) < MIN_DISTANCE then
6         move_signals_between({Signals}, object_A, object_B)
```

Result: {Signals}

La Figura 2.23 ilustra un sistema de vía simple con un paso a nivel (Lc01) y una plataforma (Plat01) para tres casos: suficiente separación entre elementos (vía superior), poca distancia entre los elementos sin aplicar el Algoritmo 15 (vía intermedia) y poca distancia entre elementos pero aplicando el Algoritmo 15 (vía inferior).

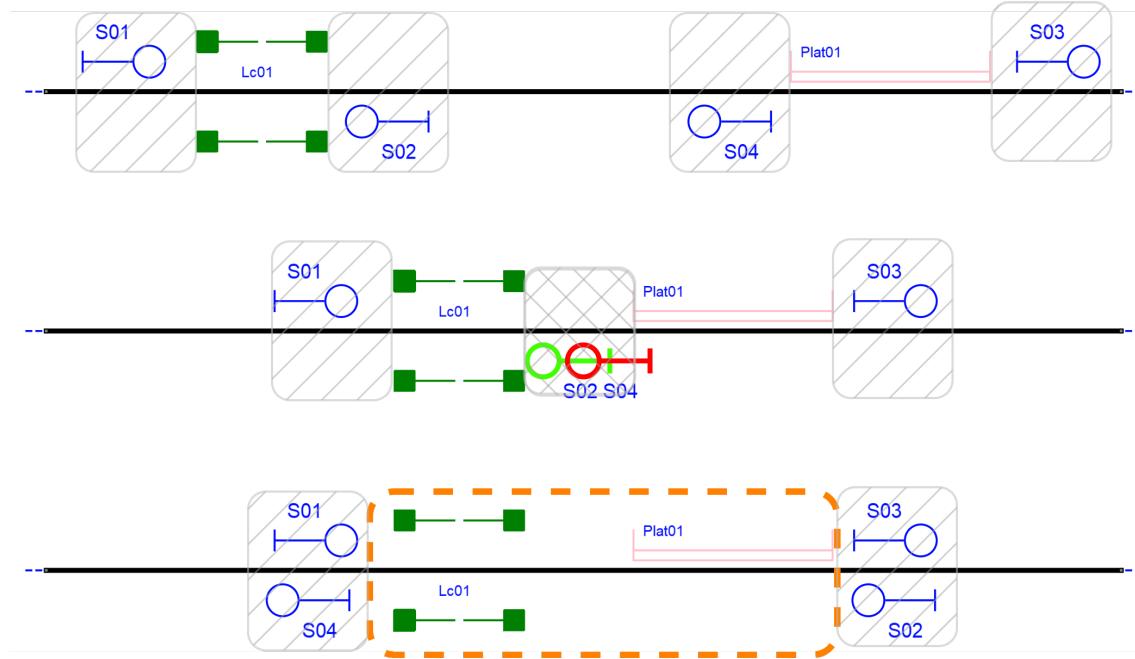


Figura 2.23: Señalamiento generado, simplificado por algoritmo de herencia horizontal.

En la vía superior se presenta el caso de un paso a nivel y una plataforma separados una distancia suficiente. Esta distancia es tal que sus regiones de señalamiento (indicadas mediante rayas grises oblicuas) no se solapan. Por lo tanto, las señales S01 y S02 que protegen el paso a nivel y las señales de partida de la plataforma S03 y S04 no interfieren entre sí.

En la vía intermedia, la distancia entre ambos elementos ferroviarios fue reducida. Las regiones de señalamiento entre el paso a nivel y la plataforma se solapan. En función de los parámetros de distancia al paso a nivel y la plataforma elegidos para el señalamiento, las señales S02 y S04 pueden estar en distinto orden. Es decir, una formación que circula de derecha a izquierda podría ver primero la señal S02 y luego la señal S04, o viceversa, en base a los parámetros de configuración elegidos. Incluso la señal S02 podría situarse sobre la plataforma y la señal S04 dentro del paso a nivel. Estos son todos casos no deseados, que alteran el funcionamiento normal de la red y van en contra de los principios de señalamiento.

En la vía inferior tenemos la misma infraestructura que en la vía intermedia, pero se aplicó el Algoritmo 15 de herencia horizontal. La señal S02 que se utilizaba para detener la formación previo a un paso a nivel en caso de que la barrera aún no estuviese baja es adelantada al comienzo de la plataforma. De esta manera, el conductor se detendrá algunos metros antes del paso a nivel. La señal S04 que se utilizaba como señal de partida de la plataforma es retrasada hasta después del paso a nivel. De esta manera, la formación partida de la plataforma cuando el aspecto de la señal S04 sea verde, pero eso no sucederá en tanto y en cuanto la barrera del paso a nivel no se encuentre baja.

La aplicación del Algoritmo 15 de herencia horizontal despeja las señales que quedaron atrapadas en el poco espacio físico entre ambos elementos ferroviarios. El resultado final es la anulación de la región de señalamiento intermedia, considerando a la entidad Lc01-Plat01 como un mismo elemento ferroviario, resaltado en el recuadro naranja de la Figura 2.23. Este nuevo elemento adopta las funcionalidades de ambos elementos individuales simultáneamente: la formación debe detenerse en la plataforma hasta que la señal de partida habilite su salida y solo se podrá atravesar el paso a nivel si la señal de circulación lo habilita previa confirmación de que la barrera se encuentra baja.

El Algoritmo 15 de herencia horizontal puede aplicarse a mas elementos, combinándolos en entidades mas grandes de ser necesario. El señalamiento resultante protegerá ambos extremos del nuevo elemento, sin sacrificar ninguna de las funcionalidades de sus miembros.

2.3.3. Limpieza de señales

El Algoritmo 15 de simplificación por herencia vertical y el Algoritmo 15 de simplificación por herencia horizontal tienen diferentes efectos en la cantidad de señales totales. El Algoritmo de herencia horizontal mueve las señales entre elementos cuando no existe suficiente espacio entre ellos. Sin embargo, el Algoritmo de herencia vertical duplica la cantidad de señales si se aplica el criterio #2 o el #4. Todos los algoritmos que utiliza el RNA pueden aplicarse de forma iterativa si las condiciones lo permiten, pero particularmente el Algoritmo 15 duplicará las señales por cada nivel de profundidad de la red ferroviaria. Esto llevará a tener mas señales que las necesarias en algunos netElements. Este exceso de señales deben eliminarse para mantener una proporción uno a uno entre una señal y una funcionalidad específica. El Algoritmo 14 se ocupa de la limpieza de las señales que se encuentran muy próximas entre sí, eliminando siempre las que tengan menor prioridad.

Algoritmo 14: Algoritmo de reducción de señales

```

1 Sources = { T,L,S,B,P,C,J,X }
2 for Sig_A & Sig_B in [Signals] do
3   if A != B & Sig_A.From == Sig_B.From then
4     Signalling_zones.Pos = detect_next_dangers( Sig_A )
5     A.Pos,B.Pos = Sig_A.Pos,Sig_B
6     Safe = zone_between(Signalling_zones.Pos,A.Pos,B.Pos)
7     if NOT safe then
8       if |A.Pos,B.Pos| < MIN_DISTANCE then
9         if Sig_A & Sig_B same orientation then
10           if Sig_A & Sig_B same direction then
11             if Sig_A & Sig_B type "Circ" then
12               if Sig_A.Idx < Sig_B.Idx then
13                 delete Sig_B else Sig_A
14
15       if Sig_A & Sig_B different source then
16         delete_lowest_priority( Sig_A,Sig_B )

```

Result: [Signals]

La prioridad es definida según el origen de cada señal. Como se explicó en la Sección 2.2, las señales pueden tener diferentes etiquetas en función de que elemento ferroviario están protegiendo. Las señales S C B y H protegen los cambios de vías, las señales P a las plataformas, las señales J a las junturas, las señales X a los cruces de vías. Todas ellas son señales de circulación, salvo las señales B y H que son de maniobras. La prioridad mas alta fue asignada a las señales que protegen los finales de vía absolutos (T) y relativos (L) porque son las únicas señales de parada que no pueden ser reemplazadas por señales menos restrictivas.

Esto no significa que todas las señales que protegen los finales de vía (T) eliminarán todas las señales de los pasos a nivel (X), sino que de tener otra señal cercana del mismo tipo que apunte en la misma dirección y sentido, pero de mayor prioridad, será ésta quien cumpla la función de proteger al paso a nivel. Las señales pueden tener diferentes roles, como ser señales de parada, de circulación, de partida o maniobra. Las señales de parada son absolutas y se utilizan en finales de vía, mientras que las señales de partida se utilizan para reiniciar la marcha de la formación.

Las primeras poseen un aspecto rojo, mientras que las segundas pueden tener hasta tres aspectos. Las señales de circulación poseen tres aspectos y permiten velocidades mas altas en las ramas principales de la red. Las señales de maniobra poseen solamente dos aspectos y son utilizadas en los desvíos, tanto en ramas divergentes como en convergentes.

En el caso de que dos señales que tengan igual dirección, sentido y origen, pero no tengan una región de señalamiento entre ellas, deban ser simplificadas. En este caso, se decidió eliminar la señal con índice mas alto. Esto se debe a que, una vez generado el señalamiento para cada elemento, las señales de mayor índice serán las producidas por los algoritmos de herencia horizontal (Algoritmo) y vertical (Algoritmo), que tiendan a añadir señales que con mucha probabilidad deben ser eliminadas a posteriori.

Es importante resaltar que la asignación de prioridades es de vital importancia para la seguridad. Por ejemplo, que las señales B tengan mayor prioridad que las señales P. Si la plataforma se encuentra lo suficientemente alejada de un cambio de vías, una señal de maniobra que protege el cambio de vías no puede reemplazar una señal de circulación que protege a una plataforma porque siempre habrá una plataforma entre ambos y, por lo tanto, una región de señalamiento. Sin embargo, si la plataforma se encuentra lo suficientemente cerca del cambio de vías, la señal de maniobras B se movería hacia la región de señalamiento donde se encuentra la señal de circulación P, debido al Algoritmo 15 de herencia horizontal por el cual dos elementos ferroviarios muy próximos combinan sus regiones de señalamiento. Por lo tanto, la señal B de dos aspectos reemplazara a la señal P de tres aspectos, reduciendo la velocidad máxima permitida en esa zona de la red ferroviaria, incrementando la seguridad de la red.

2.4. Generación de las tablas de enclavamiento

Una vez que se genera un señalamiento óptimo y simplificado, es necesario establecer las rutas ferroviarias para crear la tabla de enclavamientos. Una ruta ferroviaria es el camino mas corto entre dos señales consecutivas que apuntan en la misma dirección, usando las vías en el sentido definido. Es decir, si el único camino entre dos señales A y B es una vía que solo puede circularse de B hacia A, no existe ruta posible entre ambas señales. Una ruta puede contener máquinas de cambios, pasos a nivel, plataformas o cualquiera de los elementos ferroviarios definidos en la Sección 2.1.

Toda ruta ferroviaria se define con una señal de inicio y una señal de llegada, indicando el estado mas seguro correspondiente a cada elemento ferroviario que involucra. Por ejemplo, por el principio de infraestructura, si la ruta atraviesa un paso a nivel, debe indicarse que ese paso a nivel debe tener su barrera baja. Ademas, si la ruta atraviesa un cambio de vías, por el principio de no bloqueo, debe indicarse la posición del cambio para que la ruta sea segura. Incluso es fundamental establecer que secciones de vías atraviesa la ruta, por el principio de autoridad y granularidad.

El mecanismo por el cual el RNA detecta y asigna las rutas ferroviarias es el Algoritmo 15. Este algoritmo itera entre todas las señales asumiendo que son la señal de inicio de alguna ruta. Las señales de parada absolutas como las utilizadas para proteger los finales de vía absolutos (*bufferStops*) no pueden ser una señal de inicio y, por lo tanto, son ignoradas por el Algoritmo 15. Solamente las señales de partida, de circulación y de maniobra son consideradas como señales de inicio de potenciales rutas ferroviarias.

Algoritmo 15: Algoritmo de detección y registro de rutas ferroviarias.

```

1 Routes = []
2 route = 0
3 for start in [Signals] do
    /* Find manoeuvre and circulation signals */
    if start != "Stop" then
        dir = start.sig.Direction
        /* Find next signal w/ same direction */
        end = find_next_signal(start,[Signals])
        [paths] = find(start.node,end.node,graph)
        for node in [paths] do
            /* Find rail objects within path */
            sws = find(graph[node],switches)
            lc = find(graph[node],levelCrossings)
            ptf = find(graph[node],platforms)
            Routes[route++] = {start,end,dir,node,sws,lc,ptf}

```

Result: [Routes]

El RNA encuentra los caminos posibles a partir de la señal de inicio, recorriendo el grafo ferroviario en el mismo sentido que la señal de entrada. Cada tramo de vía plausible de ser recorrido es intentado, si se encuentra una señal de igual dirección y sentido que la inicial, se la asigna como señal de llegada y se computa la ruta, incrementando el contador de ruta. Si se llega al final de la vía sin haber encontrado una señal compatible, el camino no es considerado una ruta válida y es descartado. Una vez que se probaron todos los caminos posibles con la señal de inicio analizada, incluyendo todas las ramificaciones, entonces se procede con la siguiente señal, hasta analizarlas todas.

Por ejemplo, en la Figura 2.24 se destacan tres rutas sobre un señalamiento generado por el RNA, con la señal S22 como señal de inicio. Podemos identificar tres rutas ferroviarias: desde S22 hasta S32, desde S22 hasta X15 y desde S22 hasta T05. Una misma señal de inicio puede definir mas de una ruta ferroviaria.

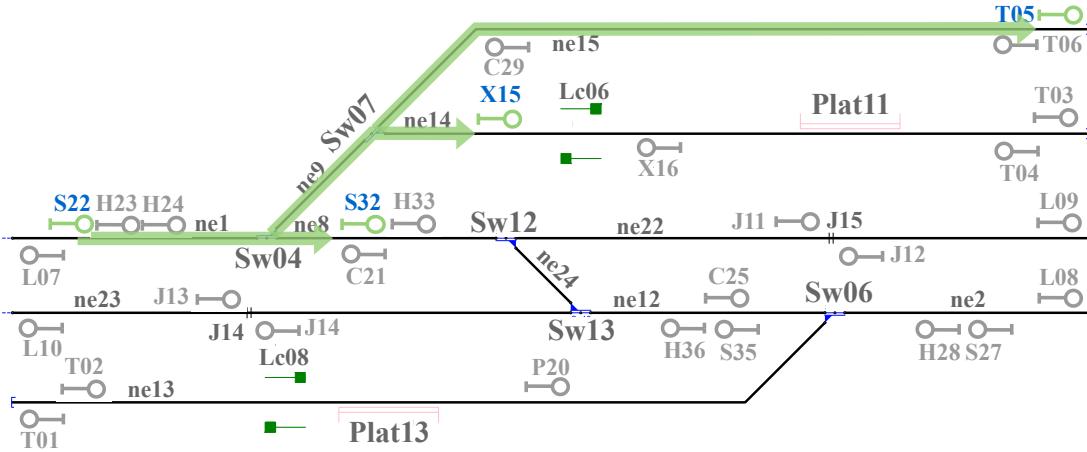


Figura 2.24: Signalling simplified in Example 1 with three routes indicated.

En paralelo a la detección de las rutas, que parte desde una señal de inicio y encuentra la señal

de llegada, se va llevando un registro de los elementos ferroviarios atravesados por estas rutas. Entre estos elementos podemos encontrar las máquinas de cambios (indicando si deben estar en posición normal o reversa), los pasos a nivel (que deben estar cerrados), plataformas y los tramos de vías utilizados. En el caso de la Figura 2.24, una ruta definida por las señales S22 y T05 recorrer los tramos de vías asociados a los *netElements* ne1, ne9 y ne15, usando el cambio de vías Sw04 en posición reversa y el cambio de vías Sw07 en posición normal, sin atravesar ningún paso a nivel.

Si dos rutas comparten las mismas señales de inicio o de llegada, estas son rutas conflictivas. De igual manera serán rutas conflictivas si comparten un tramo de vías, un cambio de vías o un paso a nivel. Esto significa que el sistema de enclavamientos no puede aprobar la solicitud de habilitación de ambas rutas simultáneamente. La aprobación de una ruta implica que la otra debe bloquearse hasta que la primera sea completada o cancelada.

Dos rutas pueden tener la misma señal de inicio y de llegada, pero haciendo uso de diferentes tramos de vías. Sin embargo, no pueden habilitarse al mismo tiempo porque, con toda seguridad, ambas rutas compartirán un cambio de vías, con posiciones habilitantes contrarias. Tal es el caso de las rutas S22-S32 y S22-X15 que ambas deben hacer uso del cambio de vías Sw04, pero mientras la primer ruta lo necesita en posición normal, la segunda lo necesita en posición reversa.

Las rutas conflictivas son detectadas por el RNA e indicadas en la tabla de enclavamientos generada automáticamente. Cada ruta es creada basada en la red de grafos ferroviaria utilizando el algoritmo de camino óptimo para grafos dirigidos. Otros artículos [65, 153, 154, 160] exploran otras estrategias para encontrar las rutas basándose en tablas de enclavamientos.

En [160] abordan la generación automática de una tabla de enclavamientos a partir de un señalamiento previamente diseñado y en sus conclusiones destacan que sería deseable a futuro hacer su sistema compatible con railML para poder realizar un análisis mas profundo y, por lo tanto, generar una tabla de enclavamientos mas precisa. En [153] generan la tabla de enclavamientos utilizando la topología pero, nuevamente, no es compatible con railML y presenta muchas restricciones para aplicar su método: el señalamiento tiene que ser dado previamente, solo se admiten señales principales, solo se admiten máquinas de cambios simples y no se admiten pasos a nivel. En [154] se valida la tabla de enclavamientos junto con el trazado de vías, pero deben ser dadas previamente por el usuario. Finalmente, en [65] se genera la tabla de enclavamientos en base a la topología, pero solo considera como únicos elementos ferroviarios a las vías y las máquinas de cambios.

El RNA, en cambio, puede generar la tabla de enclavamientos sin restringir ni el número ni el tipo de elementos ferroviarios usados. Además, valida su propia tabla de enclavamientos contra la tabla de enclavamientos original, si hubiese. También es destacable que al ser compatible con railML es posible utilizarlo en conjunto con otras herramientas compatibles con el estándar, aumentando su escalabilidad y usabilidad.

2.5. Validación del señalamiento y las tablas de enclavamiento

La validación a nivel general del señalamiento generado por el RNA se realiza comparándolo con el señalamiento original. Esta comparación se realiza mediante sus tablas de enclavamientos, que representan de forma objetiva y detallada el listado de rutas y como se relacionan con la infraestructura ferroviaria. Si no existiese señalamiento previo y, por lo tanto, no existe una tabla de enclavamientos original contra la cual comparar, entonces la validación es exitosa por defecto. Los detalles particulares respecto a la validez del señalamiento y la seguridad que aportan al sistema son discutidos en la Sección H.6.

En el caso de que exista un señalamiento previo, el RNA utiliza el Algoritmo 16 para validar el señalamiento generado. Para lo cual, en primer lugar, se genera una red de digrafos en base a la tabla de enclavamientos original. Este digrafo tendrá como nodos iniciales y finales los *netElement*

iniciales y finales de cada ruta respectivamente. Las rutas, a su vez, serán las aristas que conecten los nodos. Análogamente se crea un digrafo que represente al nuevo señalamiento generado por el RNA.

Algoritmo 16: Algoritmo de validación de tablas de enclavamiento ferroviarias.

```

1 old_graph = create_Digraph()
2 new_graph = create_Digraph()
/* Create digraphs for both interlocking tables */
for start_net,end_net,old_route in old_table do
    old_graph ⇐ start_net,end_net,old_route
for start_net,end_net,new_route in new_table do
    new_graph ⇐ start_net,end_net,new_route
/* Detect coincidences between both digraphs */
routes_found = 0
for start_net,end_net,old_route in old_table do
    if start_net != end_net then
        routes_found += find_shortest_paths(start_net , end_net)
    else
        if start_net in new_graph then
            routes_found += 1
        else
            if end_net in new_graph then
                begin_nodes = List(neighbours of end_net in new_graph)
                if begin_nodes != [] then
                    routes_found += find_shortest_paths(begin_nodes[0] , end_net)

```

Result: routes_found

A continuación, el Algoritmo 16 recorre la lista de rutas originales. Si es posible encontrar la ruta original en el nuevo digrafo, con el mismo sentido de circulación y mismos netElements de inicio y destino, entonces se computa que ambas rutas son equivalentes. Es de esperar que, por el principio de granularidad que subdivide las rutas largas en rutas mas pequeñas y funcionales, la cantidad de rutas generadas por el RNA sea igual o mayor que la cantidad de rutas originales.

Este es el caso de una ruta original que se encuentra definida entre los netElement A y D, pero no es posible encontrar rutas en el digrafo generado que cubran el trayecto A-D. Entonces se procede a calcular todas las sucesiones de rutas que, comenzando por A, puedan concatenarse, sin solaparse, hasta llegar a D. Por ejemplo, las rutas que recorran A-B, B-C y C-D serán tres rutas que, en conjunto, serán equivalentes a la ruta original A-D. Por el principio de autoridad, podemos afirmar que la combinación de rutas mas pequeñas serán equivalentes a rutas mas largas del señalamiento original y, a su vez, la combinación de todas las rutas permitirán recorrer toda la locación. Cada ruta particionada encontrada se computa como una nueva ruta encontrada. Por lo que una ruta compuesta de N rutas simples que, en conjunto, equivalen a una de las rutas originales, será computada como N rutas.

Adicionalmente, es probable que el señalamiento generado por el RNA proteja elementos ferroviarios no contemplados en el señalamiento original, añadiendo mas rutas a la tabla de enclavamientos. Estas rutas no tendrán un equivalente en el señalamiento original, pero son computadas igualmente ya que amplían la logística y seguridad de la red ferroviaria.

El grado de cobertura del señalamiento generado por el RNA será una proporción entre la cantidad de rutas generadas y las originales. Si alguna de las rutas originales no tiene un equivalente directo o compuesto con las rutas generadas por el RNA, la cobertura será menor al 100 %. Si

la cantidad de rutas es la misma y no existe ninguna ruta original sin equivalente en el nuevo señalamiento entonces la cobertura es del 100 %. Si, adicionalmente a la equivalencia total de rutas originales y generadas, el señalamiento generado incorpora nuevas rutas que incrementan la seguridad y la logística de la red, la cobertura es mayor al 100 %. La validación del señalamiento es exitosa si la cobertura iguala o supera al 100 %.

2.6. Comprobación de principios de señalamiento ferroviario

En esta sección se abordará el proceso de validación de cada uno de los principios ferroviarios establecidos en la Sección 1.2, como parte del proceso expuesto en la Sección 1.7.3. Para ello, serán abordados de manera genérica cada uno de los principios de señalamiento junto con el algoritmo que el RNA utiliza para validarlos, previo a exportar el señalamiento generado.

2.6.1. Comprobación del principio de autoridad

El principio de autoridad radica en que las rutas otorgan a las formaciones un permiso de uso limitado para circular por las vías. No obstante, la combinación de todas las autoridades disponibles de ser otorgadas deben cubrir la totalidad de la traza ferroviaria. Es decir, no debe existir ninguna sección de vía que se encuentre fuera del alcance de las rutas y, por lo tanto, aislada de la red. Puede darse el caso de que una sección de vía tenga una ruta de entrada o de salida, pero no ambas. También puede darse el caso de que la sección de vía sea transitada, pero no sea ni inicio ni final de ninguna ruta.

El RNA utiliza el Algoritmo 17 para validar el principio de autoridad comparando los netElements alcanzados por las rutas disponibles, con la lista de netElements potenciales de ser alcanzados. Esta última salvedad deja fuera las secciones de vías que preceden a una señal de circulación posterior a un final de vía relativo (lineBuffer). Esto se realiza ya que estas secciones de vía son transitadas por rutas fuera de la locación bajo análisis, rutas que terminan en la señal de circulación mencionada.

Algoritmo 17: Algoritmo de validación del principio de autoridad.

```
1 Validated = False
2 {Routes}, {netElements} without netElements with lineBuffer
/* Validate that every route covers all the railway tracks. */
for route in {Routes} do
    if route[begin] in {netElements} then
        Remove route[begin] from {netElements}
    if route[end] in {netElements} then
        Remove route[end] from {netElements}
    for track in route[paths] do
        if track in {netElements} then
            Remove track from {netElements}
if {netElements} is Empty then
    Validated = True
```

Result: Validated

Se asume inicialmente que la validación es fallida por defecto. Si al finalizar el Algoritmo 17 no existe ningún netElement remanente en la lista, se considera que la validación es exitosa.

2.6.2. Comprobación del principio de claridad

El principio de claridad radica en que el señalamiento no posee señales ambiguas. Es decir, no existen dos rutas distintas que otorguen autoridad sobre las mismas secciones de vía, en el mismo sentido, utilizando la misma infraestructura. En otras palabras, las rutas en una misma dirección y sentido no se solapan, cada una comienza donde otra ha terminado.

El RNA utiliza el Algoritmo 18 para validar el principio de claridad, generando una red de grafos dirigidos. La misma se constituye utilizando todas las rutas definidas por el RNA como aristas, y a los netElements que conectan como nodos de la red. Si se detecta que existen dos rutas R' y R'' que conectan dos nodos A y B en el mismo sentido, utilizando la misma infraestructura, entonces se dice que las rutas son ambiguas y el principio de claridad no se cumple.

Algoritmo 18: Algoritmo de validación del principio de claridad.

```
1 Validated = False
2 {Routes}
/* Validate that every netElement is reached by one route only. */
for route in {Routes} do
    begin = route[begin]
    end = route[end]
    index = route[index]
    direction = route[direction]
    digraph = create_digraph(begin,end,index,direction)
if digraph has no redundant routes then
    Validated = True
```

Result: Validated

Aunque el orden de validación de los principios de señalamiento es indistinto, la confirmación de que las señales son suficientes para alcanzar toda la red (Principio de autoridad) y de que no existen señales ambiguas (Principio de claridad) facilita las siguientes validaciones. Esto se debe a que, con estas dos comprobaciones, ya sabemos que contamos con la mínima cantidad de señales para recorrer la red de forma única, garantizando la correcta logística de la red. Los siguientes principios son necesarios para validar la seguridad del señalamiento generado por el RNA.

2.6.3. Comprobación del principio de anticipación

El principio de anticipación radica en que existe una señal previo a cada peligro, situada en la zona de señalamiento. Una zona señalamiento puede estar próxima a una curva, al final de la traza ferroviaria o en las inmediaciones de un paso a nivel, una estación, un cambio de vías o cualquier infraestructura ferroviaria a ser protegida. En otras palabras, el señalamiento debe estar distribuido de tal manera que entre dos señales que constituyen una ruta exista un único peligro o situación que quiera la atención del conductor.

El RNA utiliza el Algoritmo 19 para validar el principio de anticipación, detectando las regiones de señalamiento como zonas peligro peligrosas, producto del trazado ferroviario (curvas, finales de vía) o de la infraestructura ferroviaria (plataformas, pasos a nivel, etc.).

Algoritmo 19: Algoritmo de validación del principio de anticipación.

```
1 Validated = True, {Dangers} = None
2 {Routes}, {Track}
/* Validate that each danger is protected by the signalling. */
for curve in {Tracks} do
    {Dangers} ← ADD curve
for element in {Infraestructure} do
    {Dangers} ← ADD element
for danger in {Dangers} do
    for track going into danger do
        if track has signal AND signal points to danger then
            Continue
        else
            Validated = False
            Stop iterations
```

Result: Validated

A continuación, el Algoritmo 19 comprueba que existe una señal para cada dirección del peligro, apuntando en el sentido del mismo. Garantizando de esta forma, que previo a cada peligro inicie o finalice una ruta, en caso de que la ruta sea segura de ser transitada o no, respectivamente.

Una sola región de señalamiento que tenga una de sus entradas sin proteger por una señal es suficiente para que el Algoritmo 19 devuelva un resultado negativo. Si todas las entradas de cada región de señalamiento están protegidas por una señal, entonces el principio de anticipación se encuentra cumplido. En el caso de que los elementos ferroviarios se encuentren muy próximos, por herencia horizontal, se consideran un único elemento y sus regiones de señalamiento se simplifican como se explicó en la Sección 2.3.2.

2.6.4. Comprobación del principio de granularidad

El principio de granularidad radica en que las rutas deben ser lo suficientemente cortas como para permitir una logística flexible, pero no tan cortas como para no tener ninguna funcionalidad real. Consideramos, por lo tanto, que el señalamiento debe estar distribuido de tal forma que las rutas tengan una funcionalidad básica. Entre éstas funciones tenemos el uso de un elemento ferroviario específico (o conjunto de elementos ferroviarios, si están muy próximos) o la subdivisión de rutas cuyos tramos originales eran demasiado extensos.

El RNA utiliza el Algoritmo 20 para validar el principio de granularidad, analizando que elementos ferroviarios se encuentran contenidos en cada ruta y la distancia entre las señales que conforman la misma. Se recorre el listado de rutas generadas por el RNA, que ya establece los elementos contenidos en las rutas, las señales que las definen y la posición de las mismas. En cada ruta se analiza si existen elementos ferroviarios que justifiquen la existencia de la ruta y si el largo de la misma se encuentra entre un rango mínimo y máximo estipulado. Si alguna de estas condiciones se cumple, entonces se procede a analizar la siguiente ruta. En caso contrario, el Algoritmo 20 finaliza de forma negativa. Si todas las rutas analizadas superan el proceso, entonces el principio de granularidad ha sido validado exitosamente.

Algoritmo 20: Algoritmo de validación del principio de granularidad.

```
1 Validated = True
2 {Routes}
/* Validate that each route has a purpose and a min/max length. */
for route in {Routes} do
    if MIN > route[length] > MAX OR route[elements] == None then
        Validated = False
        Stop iterations
```

Result: Validated

Si la ruta es demasiado pequeña o demasiado larga, la validación finaliza de forma negativa. Si la ruta tiene un tamaño adecuado pero no contiene ningún elemento relevante de ser protegido, también se rechaza la validación. En caso contrario, si todas las rutas cumplen el criterio entonces se da por validado el principio de granularidad.

2.6.5. Comprobación del principio de terminalidad

El principio de terminalidad radica en que el señalamiento debe indicar, sin excepciones, el final de cada rama del trazado ferroviario, sin importar si es un final relativo u absoluto. Como se explicó en la Sección 2.1.5, tanto los finales de vía relativos como los absolutos necesitan una señal de parada que permita habilitar la salida de la formación de la locación bajo estudio. Además, los finales de vía absolutos requieren de una señal de partida para reiniciar la marcha de la formación en la dirección contraria, retomando el recorrido dentro de la traza ferroviaria.

El RNA utiliza el Algoritmo 21 para validar el principio de terminalidad, recorriendo cada una de las rutas generadas y analizando aquellas que culminan o empiezan en un final de vía. Una vez detectadas las rutas que posean lineBorders o bufferStops, se procede a analizar qué clase de señales poseen. Si son compatibles con las condiciones indicadas en la Sección 2.1.5, entonces se continúa analizando con la siguiente ruta. Caso contrario, la validación termina de forma negativa. Si todas las rutas analizadas superan el proceso, entonces el principio de terminalidad ha sido validado exitosamente.

Algoritmo 21: Algoritmo de validación del principio de terminalidad.

```
1 Validated = True
2 {Routes}
/* Validate that each lineBorder and BufferStop have protective signals */
for route in {Routes} do
    if lineBorder in route[element] and route goes to lineBorder then
        if signal is not route[end] then
            Validated = False
    if bufferStop in route[element] then
        if route goes to bufferStop and signal is not route[end] then
            Validated = False
        if route start from bufferStop and signal is not route[begin] then
            Validated = False
```

Result: Validated

Este principio es el más sencillo de validar positivamente ya que, como fue diseñado el RNA, tanto los *lineBorder* y los *bufferStop* siempre tendrán señalamiento y dichas señales tienen la máxima prioridad, por lo que no pueden simplificarse con otras señales. No obstante, si el usuario elige de forma explícita no generar señales para estos elementos ferroviarios, entonces este principio

de señalamiento podría no cumplirse.

2.6.6. Comprobación del principio de infraestructura

El principio de infraestructura radica en que el señalamiento debe proteger, sin excepciones, toda infraestructura ferroviaria crítica, tales como las plataformas y los pasos a nivel. Con excepción de los cambios de vías, que son cubiertos por el principio de no bloqueo. La infraestructura deberá estar protegida por el señalamiento tal como se explicó en la Sección 2.1.7 y la Sección 2.1.8, permitiendo a las formaciones detenerse en las plataformas y antes de los pasos a nivel, respectivamente.

El RNA utiliza el Algoritmo 22 para validar el principio de infraestructura, recorriendo cada una de las rutas generadas y analizando aquellas que atraviesan paso a nivel o que inician o finalizan en una plataforma.

Algoritmo 22: Algoritmo de validación del principio de infraestructura.

```
1 Validated = True
2 {Routes}
  /* Validate that each platform and crossing have protective signals */
  for route in {Routes} do
    if platform in route[element] then
      if route goes across platform and signal is not route[end] then
        Validated = False
      if route start from platform and signal is not route[begin] then
        Validated = False
    if levelCrossing in route[element] then
      if route goes to levelCrossing and signal is not route[end] then
        Validated = False
      if route start before levelCrossing and signal is not route[begin] then
        Validated = False
  Result: Validated
```

Una vez detectadas las rutas, se procede a analizar que clase de señales poseen. Si son compatibles con las condiciones indicadas en la Sección 2.1.7 y la Sección 2.1.8, entonces se continúa analizando con la siguiente ruta. Caso contrario, la validación termina de forma negativa. Si todas las rutas analizadas superan el proceso, entonces el principio de infraestructura ha sido validado exitosamente.

En el caso de que las plataformas y los pasos a nivel se encuentren muy próximos, por la simplificación de señalamiento por herencia horizontal explicada en la Sección 2.3.2, igualmente el Algoritmo 22 buscará las señales mas próximas al nuevo elemento ferroviario compuesto. De esta manera, al analizar todas las rutas exitosamente, es posible validar el señalamiento que protege a los elementos ferroviarios críticos, cumpliendo el principio de infraestructura.

2.6.7. Comprobación del principio de no bloqueo

El principio de no bloqueo radica en que el señalamiento debe proteger los cambios de vías en sus tres direcciones: inicio, normal y desvío, como se explica en la Sección 2.1.9. A su vez, debe garantizarse que los cambios compuestos no tengan señales intermedias, para evitar que las formaciones se detengan en zonas críticas, bloqueando los cambios de vías para otras formaciones que las requieran, como se explica en la Sección 2.3.1.

El RNA utiliza el Algoritmo 23 para validar el principio de no bloqueo, recorriendo la lista de máquinas de cambios. El primer paso es agrupar en tres listas distintas todos los netElements de cada cambio de vía según su dirección: inicio, normal y desvío. Luego, son removidas de las listas de inicio y normal todos los netElements que también pertenezcan a la lista de desvío. De esta manera, el algoritmo contempla a los cambios compuestos productos de la simplificación por herencia vertical (ver Sección 2.3.1).

Algoritmo 23: Algoritmo de validación del principio de no bloqueo.

```
1 Validated = False
2 {Switches}
3 switch_start = [], switch_normal = [], switch_branch = [], critical = []
/* Validate that each switch have protective signals */
/* List netElements for each side of each switch */
for sw in {Switches} do
    switch_start ← ADD sw[netElement] for 'Start'
    switch_normal ← ADD sw[netElement] for 'Normal'
    switch_branch ← ADD sw[netElement] for 'Branch'

/* Remove netElements that also belongs to a branch */
for sw in {Switches} do
    switch_start ← REMOVE netElement if in switch_branch
    switch_normal ← REMOVE netElement if in switch_branch
    critical ← ADD netElement in switch_branch AND (switch_start OR switch_normal)

/* Remove the netElements that have signals */
for netElement that has a signal do
    if netElement in unprotected_switch_start then
        switch_start ← REMOVE netElement
    if netElement in unprotected_switch_normal then
        switch_normal ← REMOVE netElement

/* Final condition */
if switch_start == [] AND switch_normal == [] AND critical has no signals then
    Validated = True
```

Result: Validated

A continuación, se remueven de las listas de inicio y normal todos los netElements que tienen una señal apuntando hacia el cambio de vías. Finalmente, si ambas listas se encuentran vacías y los netElements de la lista de cambios que también pertenezcan a las listas de inicio y normal originales no tienen señales, entonces el principio de no bloqueo ha sido validado exitosamente.

Una vez que todos los principios de señalamiento ferroviario han sido validados podemos afirmar que el señalamiento generado por el RNA es seguro y cumple con todos los requisitos planteados en la Sección 1.2. Dicho señalamiento deberá ser implementado por el Generador Automático de Códigos en VHDL (ACG), explicado en profundidad en la Sección 3.

Capítulo 3

Automatic Code Generator (ACG)

Una vez que el RNA finaliza el procesamiento de los elementos estáticos y dinámicos de la locación ferroviaria, se obtiene una representación de la misma tanto en formato railML como en redes de grafos. El Automatic Code Generator (ACG) utiliza la red de grafos para implementar el sistema de enclavamiento en formato VHDL, compatible con la tecnología FPGA (del inglés, Field Programmable Gate Arrays). En esta sección se abordarán las funcionalidades del sistema de enclavamientos implementadas por el ACG (Sección 3.1) y la arquitectura general del sistema generado (Sección 3.2).

Además, se profundizará en la arquitectura y modelado dinámico de los módulos de comunicación, detección de tramas, decodificación de tramas, codificación de tramas, impresión de resultados, selección de modos de funcionamiento y red ferroviaria. Dentro del módulo de red ferroviaria se hará especial énfasis en la arquitectura y las redes de Petri que modelan el comportamiento dinámico de los cruces de vía, cambios de vía simples, cambios de vía dobles, cambios de vía en tijeras, trazado ferroviario, señales y las rutas de la red.

Finalmente, se explicarán las redundancias implementadas para robustecer el sistema de enclavamientos (Sección 3.3) y una introducción a la interfaz gráfica generada para operar el sistema de enclavamientos (Sección 3.4).

3.1. Sistemas de enclavamiento

Tal cómo fue definido en la Sección 1.1, un sistema de enclavamientos debe gestionar las rutas ferroviarias, utilizando el señalamiento para otorgarle al conductor ferroviario autoridad o no sobre ciertas secciones de la red ferroviaria. Las decisiones se toman en base al estado actual de los elementos ferroviarios que componen la red y considerando qué estado garantiza una mayor seguridad, al impedir descarrilamientos y colisiones.

Existen diversas funcionalidades que complementan al sistema de enclavamientos. Algunas centradas en incrementar la seguridad general del sistema y otras en flexibilizar la logística de la asignación de rutas. En esta sección se describirán seis de las funcionalidades principales del sistema de enclavamientos implementadas por el ACG.

3.1.1. Bloqueo de máquina de cambios por ocupación

Evitar el descarrilamiento de las formaciones es una de las funciones del sistema de enclavamientos. Esto puede ocurrir principalmente en dos situaciones: formaciones circulando a alta velocidad en las curvas o conmutaciones en la máquina de cambios mientras una formación circula sobre

el cambio de vías. Para evitar este último escenario, el sistema de enclavamientos implementa un bloqueo de la máquina de cambios por ocupación, tal como se ilustra en la Figura 3.1.

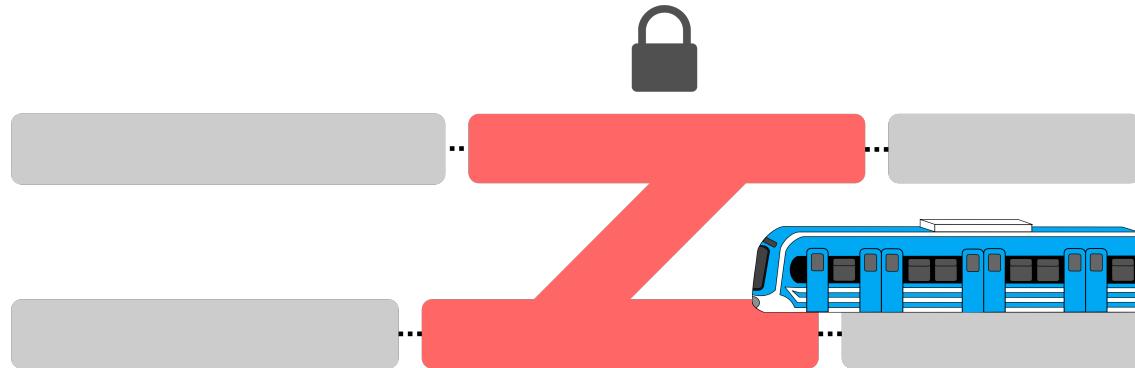


Figura 3.1: Bloqueo de máquina de cambios por ocupación de secciones adyacentes.

La funcionalidad implementada radica en inhibir la comutación de la máquina de cambios si alguna de las secciones de vías próximas al cambio de vías se encuentra ocupada. De esta manera, se garantiza que la posición del cambio de vías se mantendrá al detectar una formación aproximándose y no se permitirá su comutación hasta que la formación se encuentre completamente alejada una distancia de seguridad.

3.1.2. Requerimiento de rutas y bloqueo de cambios en ruta

Para evitar colisiones entre formaciones que circulan en sentido contrario, es necesario asegurarse que las rutas habilitadas no comparten secciones de vías o elementos ferroviarios entre sí. Para lograr esto, el sistema de enclavamientos bloquea la activación de rutas conflictivas, tal como se ilustra en la Figura 3.2.

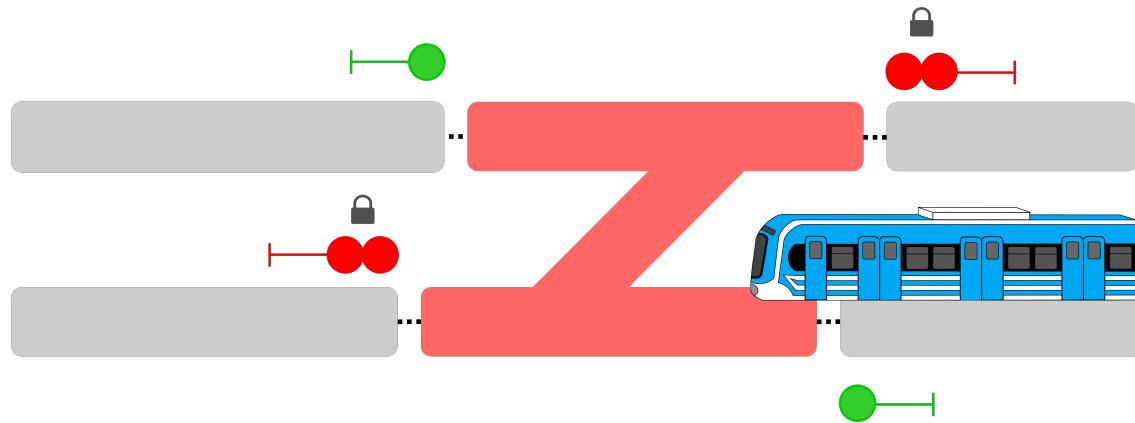


Figura 3.2: Bloqueo de rutas conflictivas.

En el ejemplo de la Figura 3.2 se puede ver una formación que circula de derecha a izquierda por la vía inferior, al tener una señal verde que la habilita. El bloqueo de rutas conflictivas se manifiesta al forzar el aspecto rojo de las señales que habilitan dichas rutas, y al inhibir cualquier cambio de aspecto posible. Las rutas conflictivas pueden compartir todo el trayecto con la ruta principal, como la señal roja de la vía inferior; pero también pueden ser señales de rutas convergentes como

la señal roja de la vía superior. De esta manera, se disminuyen las chances de colisiones frontales o laterales, respectivamente.

3.1.3. Protección por aproximación en cancelación de ruta

La distancia de frenado es un aspecto esencial a considerar cuando una ruta en curso es cancelada. La ruta en cuestión debe seguir protegida durante un tiempo de seguridad.

La Figura 3.3 introduce el caso de una formación que tenía una ruta aprobada (aspecto verde) que comenzaba en la sección violeta y abarcaba toda la sección naranja. Por seguridad, ambos cambios de vías fueron bloqueados y sus respectivas señales contrarias fueron forzadas a aspecto rojo y bloqueadas.

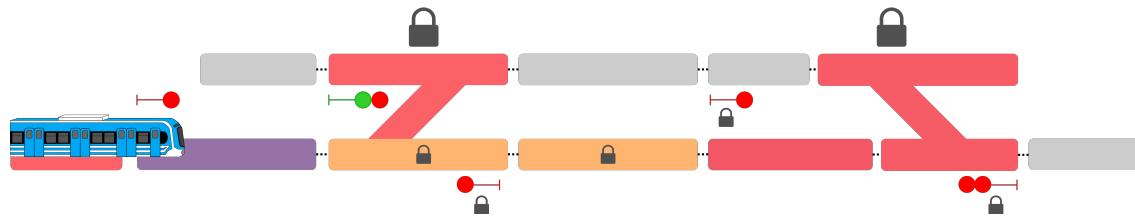


Figura 3.3: Formación aproximándose al inicio de la ruta.

Mientras la formación se encuentra en movimiento, el operador solicitó la cancelación de la ruta, cambiando el aspecto de la señal a rojo, tal como se visualiza en la Figura 3.4. La formación quizás no tenga el tiempo ni la distancia suficiente para detenerse antes de la señal, por lo que se presentan dos escenarios.

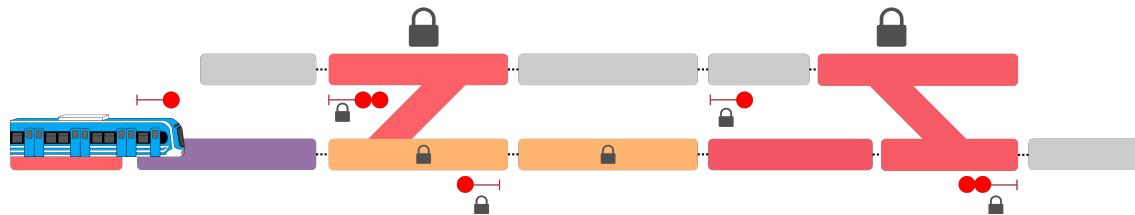


Figura 3.4: La ruta es cancelada mientras la formación se aproxima.

El primer escenario es el más favorable: la formación se detiene previo a la señal de peligro y se inicia un contador. Al cumplirse el tiempo de seguridad, las secciones asociadas a la ruta cancelada son liberadas. Lo mismo sucede con los cambios de vías y las señales conflictivas. Este tiempo otorgado permite comprobar que la formación efectivamente se detuvo antes de proceder con la liberación de los elementos ferroviarios para que puedan ser utilizados por otra ruta.

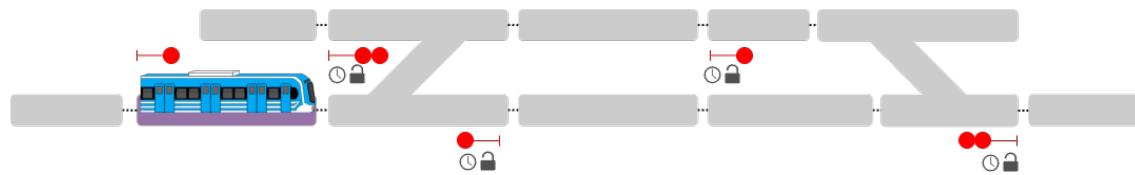


Figura 3.5: La formación se detiene exitosamente previo a la señal de peligro.

En el segundo escenario, la formación no logra detenerse previo a la señal de peligro, tal como se ilustra en la Figura 3.6. Entonces, el sistema de enclavamiento no solamente no libera las secciones

pertenecientes a la ruta cancelada, sino que también bloquea las próximas secciones, al no poder estimar cual será la distancia final de frenado de la formación.

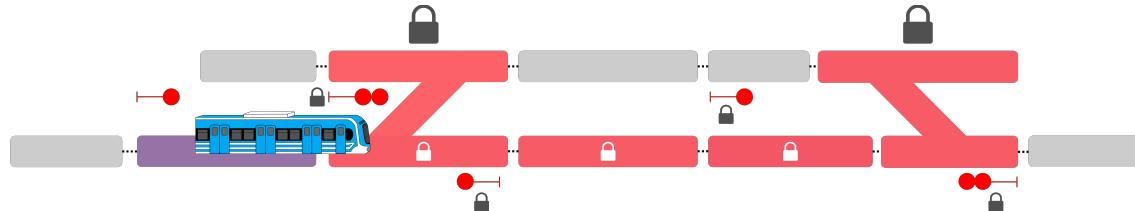


Figura 3.6: La formación no se detiene previo a la señal de peligro.

Las secciones y elementos ferroviarios próximos se mantienen protegidos y enclavados hasta que la ruta se concluya, aún habiendo sido cancelada. Al finalizar la ruta, el sistema de enclavamiento liberará las secciones y elementos ferroviarios próximos al comprobarse que la formación se detuvo previo a la señal de finalización de la ruta.

3.1.4. Protección por solape

Si una formación no detiene su marcha antes de una señal de peligro, el sistema de enclavamiento debe bloquear las secciones pertenecientes a esa ruta y la próxima, junto con la infraestructura asociado. La Figura 3.7 ilustra esta situación, donde una formación ingresa a la sección violeta pasando una señal a peligro, sin tener la autorización requerida. A diferencia de la protección por aproximación, donde una formación no logra detenerse antes de ingresar a una ruta cancelada con poca anticipación, la protección por solape se ocupa de proteger la infraestructura en el caso de que la formación ingrese a una ruta que jamás fue habilitada.

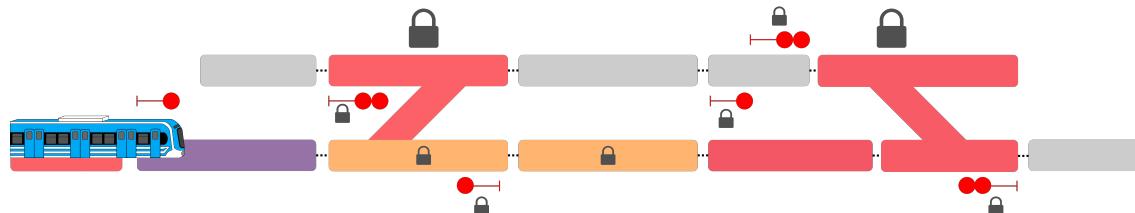


Figura 3.7: Formación ignora señal a peligro y se activa la protección por solape.

Automáticamente, las secciones de la próxima ruta (coloreadas en naranja) son bloqueadas, a la vez que los cambios de vías cercanos y todas las señales tanto consecutivas como contrarias o convergentes. El bloqueo se removerá una vez que la formación se detenga en la próxima señal a peligro, luego de un tiempo de seguridad.

3.1.5. Doble recubrimiento

Para evitar que una formación colisione con otra formación que se encuentre detenida más adelante en la misma vía o circulando a menor velocidad, el sistema de enclavamiento deberá controlar las señales entre ambas para regular la velocidad y distancia entre ellas. Tal como se explicó en la Sección 2.1.10, las señales pueden presentar diferentes aspectos. Cada aspecto determinará un rango de velocidad permitido, siendo rojo el más restrictivo. La Figura 3.8 ilustra el comportamiento del señalamiento cuando dos formaciones circulan en el mismo sentido, separadas por una distancia de seguridad.

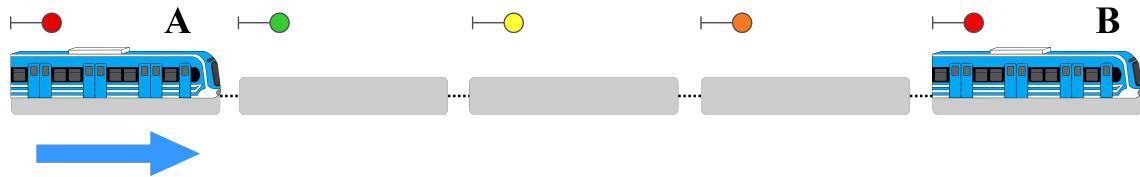


Figura 3.8: Protección por doble recubrimiento.

Debido al bloqueo por ocupación, todas las secciones ocupadas por una formación presentan una señal a peligro (roja). Inmediatamente detrás de cada formación se genera una secuencia de señales denominada doble recubrimiento. La cantidad de señales y la secuencia de aspectos variará según el operador de la red, las normas locales o nacionales. Algunos países utilizan la secuencia rojo-doble amarillo-amarillo-verde y esta es la secuencia de aspectos que implementa el ACG en este trabajo, aunque cabe aclarar que el ACG puede modificarse para implementar otras secuencias. Para simplificar la representación, en la Figura 3.8 se reemplazó la señal doble amarilla por una señal naranja. En este trabajo siempre se representará mediante una señal naranja a una señal doble amarilla.

La formación que circula por detrás (formación A en la Figura 3.8) se encuentra frente a una señal de aspecto verde, por lo que puede continuar su marcha sin restricciones, siempre y cuando su velocidad sea menor a la velocidad máxima permitida en la red ferroviaria. Si la formación A reduce la distancia a la formación B, pasará a estar regida por una señal de aspecto amarillo. Si esto sucediera, la formación A deberá disminuir su velocidad para volver a situarse dentro de una sección verde. Lo mismo ocurriría si la formación A alcanzara una señal de aspecto doble amarillo, indicada mediante la señal naranja en la Figura 3.8. En este caso dado que la distancia entre formaciones es aún menor, deberá reducirse aún más la velocidad.

Si la formación continúa teniendo una mayor velocidad que su par, la distancia entre ambas se reducirá y las señales permitirán velocidades mas o mas reducidas, hasta que la distancia se incremente a un valor seguro.

3.1.6. Liberación secuencial

Las rutas conflictivas no pueden ser habilitadas a la vez, pero existen algunas rutas que son parcialmente conflictivas solamente, porque comparten una parte de la infraestructura y no toda. La implementación de la liberación secuencial aumenta la flexibilidad en la asignación y habilitación de rutas, mejorando la logística permitida por el sistema de enclavamientos. En la Figura 3.9 se ilustra una formación que iniciará una ruta ya habilitada, para lo cual ya han sido bloqueadas las secciones (coloreadas en naranja) y la infraestructura (coloreada en rojo).

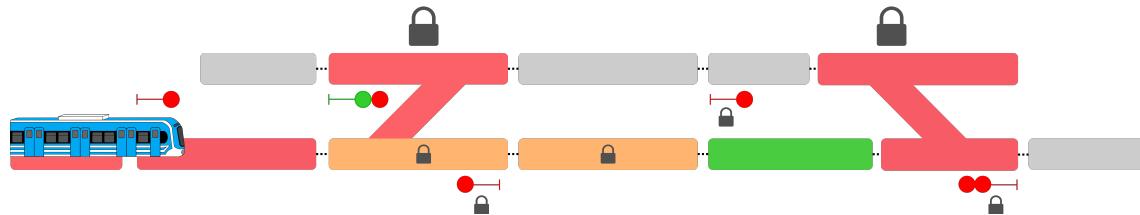


Figura 3.9: Formación iniciando una ruta ferroviaria.

Al ocupar las secciones de vías, debido al bloqueo por ocupación, la señal de inicio de la ruta pasa a peligro y se bloquea la sección consecutiva a la ruta (coloreado en naranja), debido a la protección por solape. Esto se ilustra en la Figura 3.10.

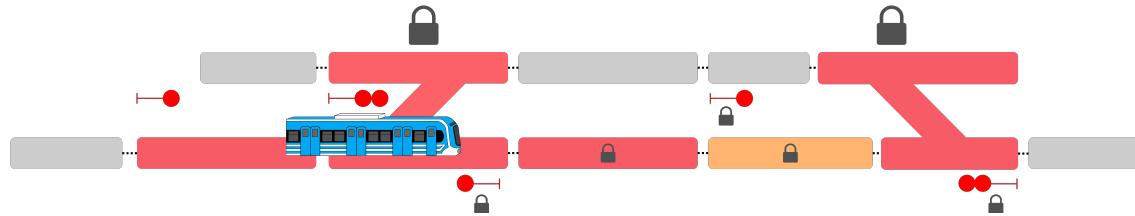


Figura 3.10: Formación activando el bloqueo por ocupación y el bloqueo por solape.

Una vez que la formación desocupa las secciones de vías asociadas al cambio de vías anterior, el sistema de enclavamientos libera inmediatamente toda la infraestructura asociada, como se ilustra en la Figura 3.11. A la vez, el sistema de enclavamientos debe esperar a que se cumpla el plazo de seguridad antes de liberar la infraestructura posterior al fin de la ruta. Solamente son liberadas las secciones y señales que ya no son conflictivas.

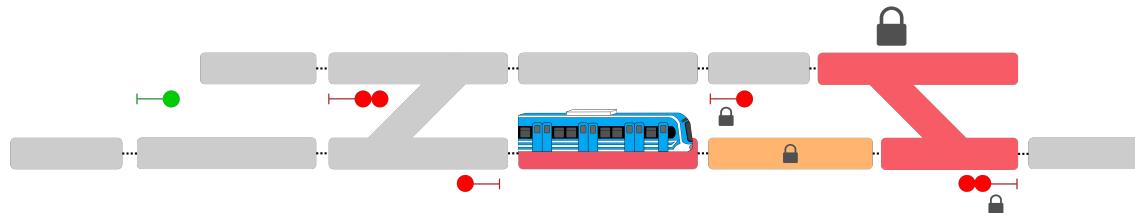


Figura 3.11: Liberación secuencial de la infraestructura por detrás de la formación.

Transcurrido el tiempo de seguridad, el sistema de enclavamientos libera las secciones, cambios de vías, señales y toda infraestructura posterior al fin de la ruta, tal como se ilustra en la Figura 3.12.

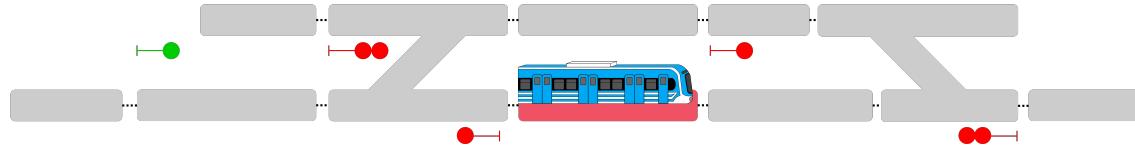


Figura 3.12: Liberación secuencial de la infraestructura por delante de la formación.

Para cada sistema de enclavamientos generado, dependiendo de la infraestructura disponible, el ACG implementa las funciones de seguridad presentadas en la Sección 3.1.1, Sección 3.1.2, Sección 3.1.3, Sección 3.1.4, Sección 3.1.5 y Sección 3.1.6.

En las siguientes secciones se profundizará en la implementación de cada uno de los módulos del sistema y su comportamiento dinámico.

3.2. Arquitectura del sistema

Para la implementación del sistema se consideró el uso de máquinas de estado finitas (FSM, del inglés Finite State Machine) o máquinas de estado finitas con camino de datos (FSMD, del inglés Finite State Machine with Data path), que son máquinas de estado finitas y circuitos secuenciales y combinacionales que constituyen el camino de datos. Se optó por utilizar FSMD en lugar de una FSM porque una FSMD aporta un mayor control del diseño a bajo nivel, una mayor portabilidad

y un uso mas eficiente de los recursos de la plataforma electrónica. La contra de usar FSMD es que su complejidad es mayor y, por lo tanto, el tamaño del sistema se incrementa.

Una FSMD, como la ilustrada en la Figura 3.13, posee dos partes diferenciadas: el camino de control y el camino de datos. El camino de control se compone de 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.

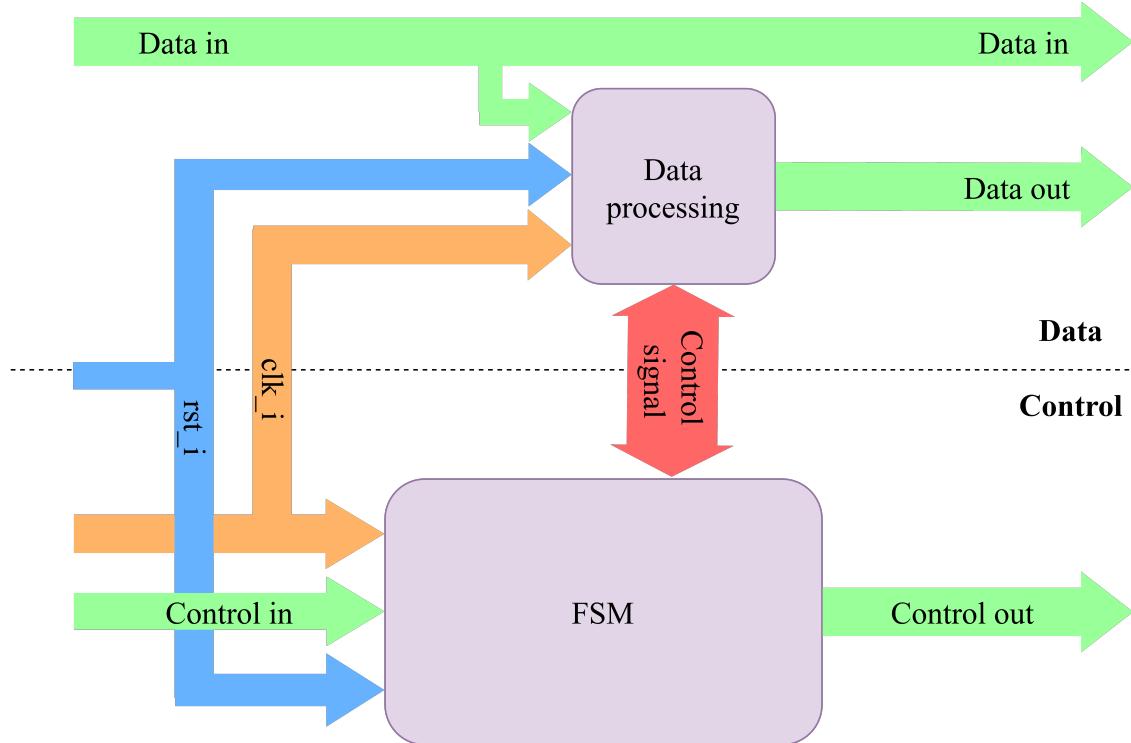


Figura 3.13: Diagrama en bloques de una FSMD genérica.

El ACG generará automáticamente cada FSMD en función de los elementos detectados en la red de grafos. Es decir, el modelado de cada elemento será una plantilla que contemplará todos los casos posibles, con todas las entradas y salidas posibles, pero solo se implementarán las funcionalidades que cada elemento particular requiera. De esta manera, será posible optimizar los recursos a la vez que para hacer la validación del sistema alcanzará con validar cada elemento una única vez, como un objeto genérico.

Los elementos ferroviarios a ser modelados por el ACG son los que denominaremos elementos dinámicos. Es decir, todo elemento ferroviario que posea algún estado susceptible de modificarse en función de algún evento o del tiempo. Los elementos dinámicos a implementar, con sus respectivos estados, son los siguientes:

- Circuitos de vía: ocupados, desocupados.
- Rutas: no solicitadas, solicitadas, reservadas, enclavadas, activadas, canceladas, expiradas.
- Señales: rojo, doble amarillo, amarillo, verde
- Pasos a nivel: brazo de la barrera bajo, brazo de la barrera alto.

- Cambios de vías simples: posición normal, posición reversa.
- Cambios de vías dobles: posición doble normal, posición doble reversa, posición normal reversa, posición reversa normal.
- Cambios de vías en tijeras: posición normal, posición reversa.

Internamente, el sistema contemplará que algunos elementos pueden admitir estados de transición. Estos estados son producto del tiempo que requiere un actuador para completar las comandos que la FPGA envía. No obstante, los estados de transición solo serán tolerados un tiempo determinado, pasado el cuál se asumirá que la orden no fue completada y se abortará la ejecución de la ruta solicitada.

La arquitectura general del sistema generado por el ACG se ilustra en la Figura 3.14. El módulo *Detector* recibe las tramas en formato serie, comprueba su integridad y en caso de que los datos contengan solamente caracteres válidos los entrega al módulo *Decoder*. El módulo *Decoder* toma esos datos y los paralleliza para entregarlos al módulo *Network* en la entrada que corresponda a cada dato. El módulo *Network* es la implementación de la red de grafos generada por el RNA.

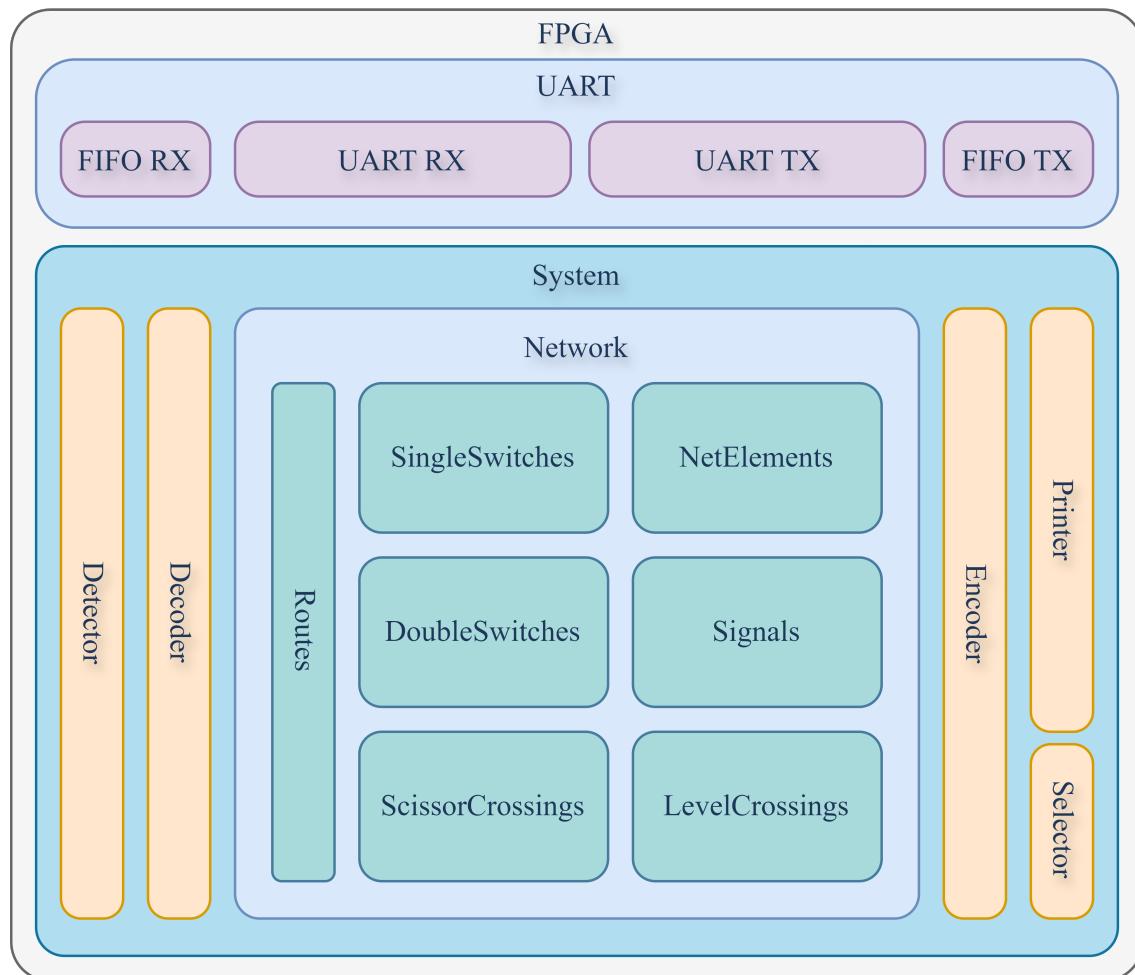


Figura 3.14: Arquitectura general del sistema generado por el ACG.

El módulo *Encoder* toma las salidas generadas por el módulo *Network*, las agrupa apropiadamente y se las suministra al módulo *Printer*. El módulo *Printer* transforma cada elemento de la señal en un carácter imprimible para enviar a la UART. Finalmente, el módulo *Selector* se usa exclusivamente a los fines de comprobar el correcto funcionamiento de la comunicación serie, puenteando a todos los otros módulos.

La descripción del sistema se realizará desde sus módulos mas externos, que implementan la comunicación del sistema: los módulos de UART, Detector, Decoder, Encoder y Printer. De forma tal de entender en alto nivel como es el proceso de comunicación hacia y desde la FPGA, para luego abordar el núcleo del sistema de enclavamiento. cuya complejidad es mucho mayor.

3.2.1. Modulo UART

Si consideramos la lista de elementos dinámicos y cada estado que pueden admitir, es claro que la cantidad de señales sobrepasaría por mucho la limitada cantidad de puertos que una FPGA pueda proveer. Por ejemplo, si se considera un sistema ferroviario como el presentado en la Figura 1.1, que por comodidad para el lector se copia en la Figura 3.15, sería necesario mas de 25 pines de la FPGA entre entradas y salidas, lo que implicaría que para ese sistema tan simple se requeriría utilizar una FPGA de tamaño medio o grande. Esto implica que ese enfoque dificultaría la implementación del sistema de enclavamiento de redes ferroviarias de mayor tamaño.

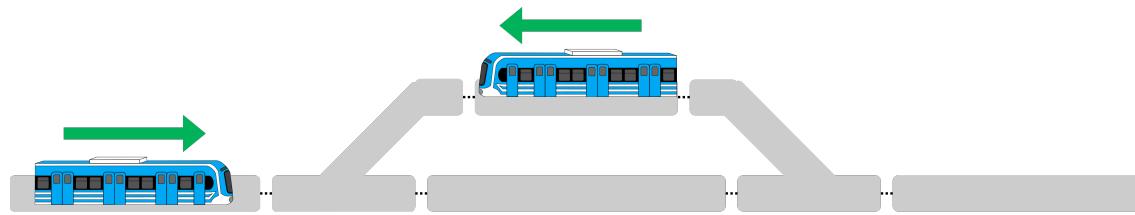


Figura 3.15: Topología de derivación ferroviaria.

Es por eso que se decidió que la FPGA mediante la cual se implemente el sistema de enclavamiento debe recibir y transmitir la información a la cabina de señalamiento en formato serie. El uso de comunicación serie para la implementación del sistema es apropiado, ya que otros sistemas ferroviarios utilizan comunicación serie, como por ejemplo RS-485 o MVB en las redes de comunicación de trenes (TCN, del inglés *Train Communication Network*) [245]. La comunicación a implementar entre el sistema de enclavamiento y la cabina de señalamiento deberá ser flexible para ser utilizada en diferentes implementaciones con menor o mayor cantidad de elementos ferroviarios.

En la Figura 3.16 se presenta la propuesta de conexión de la FPGA con una computadora externa, que para el desarrollo y prueba de la solución hará las veces de la cabina de señalamiento. En la Figura 3.16 también se representan los módulos internos de comunicación. La UART (del inglés *Universal Asynchronous Receiver-Transmitter*), junto con las memorias FIFO (del inglés *First-In First-Out*), se utilizan para implementar el intercambio de las tramas de datos entre la FPGA y la computadora.

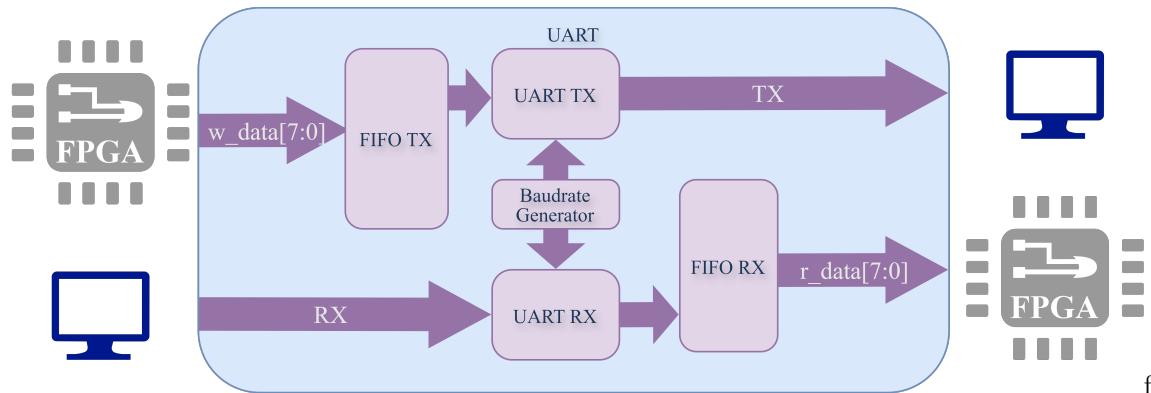


Figura 3.16: FPGA.

El módulo de recepción (UART RX), que se ilustra en la Figura 3.16, es el encargado de procesar cada bit recibido con un baudrate preestablecido y almacenar cada bit en la FIFO RX. Al completarse un byte de datos, será enviado al sistema de enclavamientos, junto con una serie de pulsos para indicar cuándo deben ser leídos. El sistema de enclavamientos esperará a tener la cantidad de bytes necesarios (definidos por el ACG) para empezar a procesar la trama. Luego, el sistema de enclavamientos devolverá una nueva trama de bytes a la FIFO TX. Finalmente la nueva trama será enviada al módulo de transmisión (UART TX) que enviará la información bit a bit, con el mismo baudrate que fue recibido.

En la Figura 3.17 se ilustra el formato definido para las tramas de entrada y salida. La trama tendrá un tamaño de entrada N y de salida M, con N igual que M. Además, la trama tendrá un carácter delimitador de entrada y de salida (< y > respectivamente). Todos los elementos de la trama serán hexadecimales en formato ASCII, para poder ser interpretados fácilmente en una terminal y ser menos susceptibles a errores por alteraciones en algún bit aleatorio.

N_NET: # netElements N_RTS: # routes N_SIG: # signals N_LCB: # levelCrossings
 N_SSW: # singleSwitches N_DSW: # doubleSwitches N_SCR: # scissorCrossings

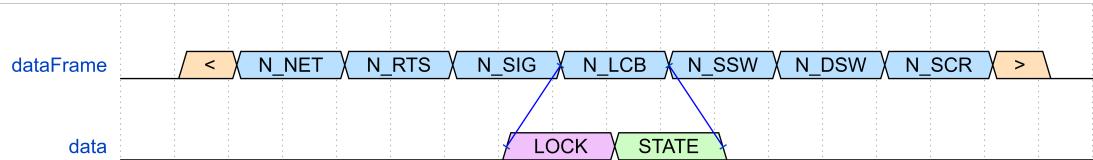


Figura 3.17: Tramas de datos y paquete de datos.

El largo de la trama de entrada y de salida queda definido por la Ecuación 3.1.

$$N = \text{1byte}(N_{NET} + N_{RTS} + N_{LCB} + N_{SSW} + N_{SCR} + N_{SIG} + N_{DSW}) \quad (3.1)$$

Cada elemento dinámico requiere un sólo carácter hexadecimal para definir su estado utilizando sus 4 bits. Los 2 bits menos significativos definen el estado (*STATE* en Figura 3.17). Por ejemplo, la posición de un cambio de vías o el aspecto de una señal. Los 2 bits más significativos definen el enclavamiento del elemento dinámico (*LOCK* en la Figura 3.17).

El parámetro *Lock* puede tomar tres valores: '00' para elementos disponibles, '01' para elementos que han sido reservados por una ruta pero aún no enclavados y '10' para elementos enclavados.

La implementación de los módulos de transmisión y recepción de la UART es invariante para cada ubicación, es decir, los recursos asignados serán los mismos, cualquiera sea el tamaño del sistema.

a implementar. Los módulos de memorias FIFO, en cambio, dependen de las características y del tamaño del sistema. Locaciones mas complejas tendrán valores de N y M mayores y, por lo tanto, requerirán FIFOs mas grandes.

3.2.2. Módulo Detector

El módulo *detector* (ver Figura 3.14) es el encargado de detectar el inicio y final de cada trama. Esto lo hace a partir de validar que el contenido de la misma tenga N caracteres, conforme al formato de trama expuesto en la Sección 3.2.1. A medida que la validación tiene lugar, los caracteres ASCII son convertidos en valores booleanos (*std_logic*) dentro de un vector de elementos booleanos llamado *packet[N]* (*std_logic_vector* de N elementos). El diagrama de bloques de la máquina de estados finitos con camino de datos se muestra en la Figura 3.18.

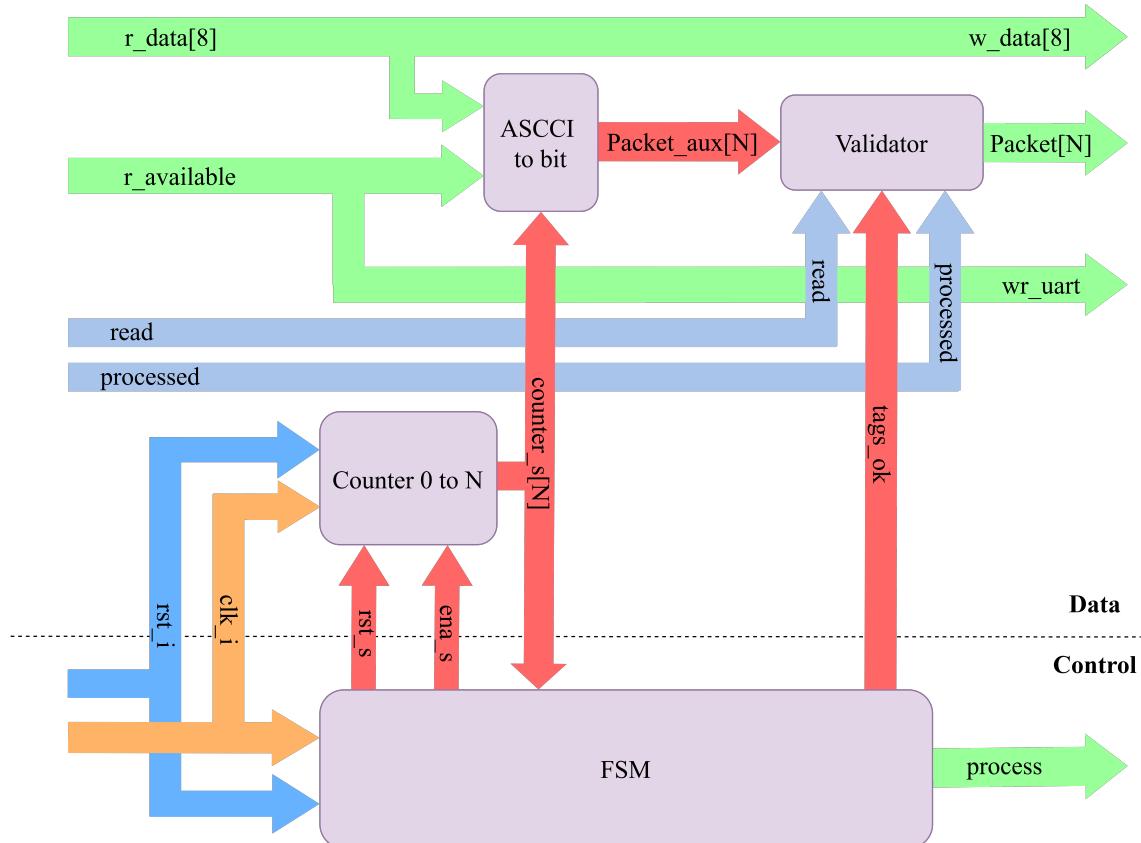
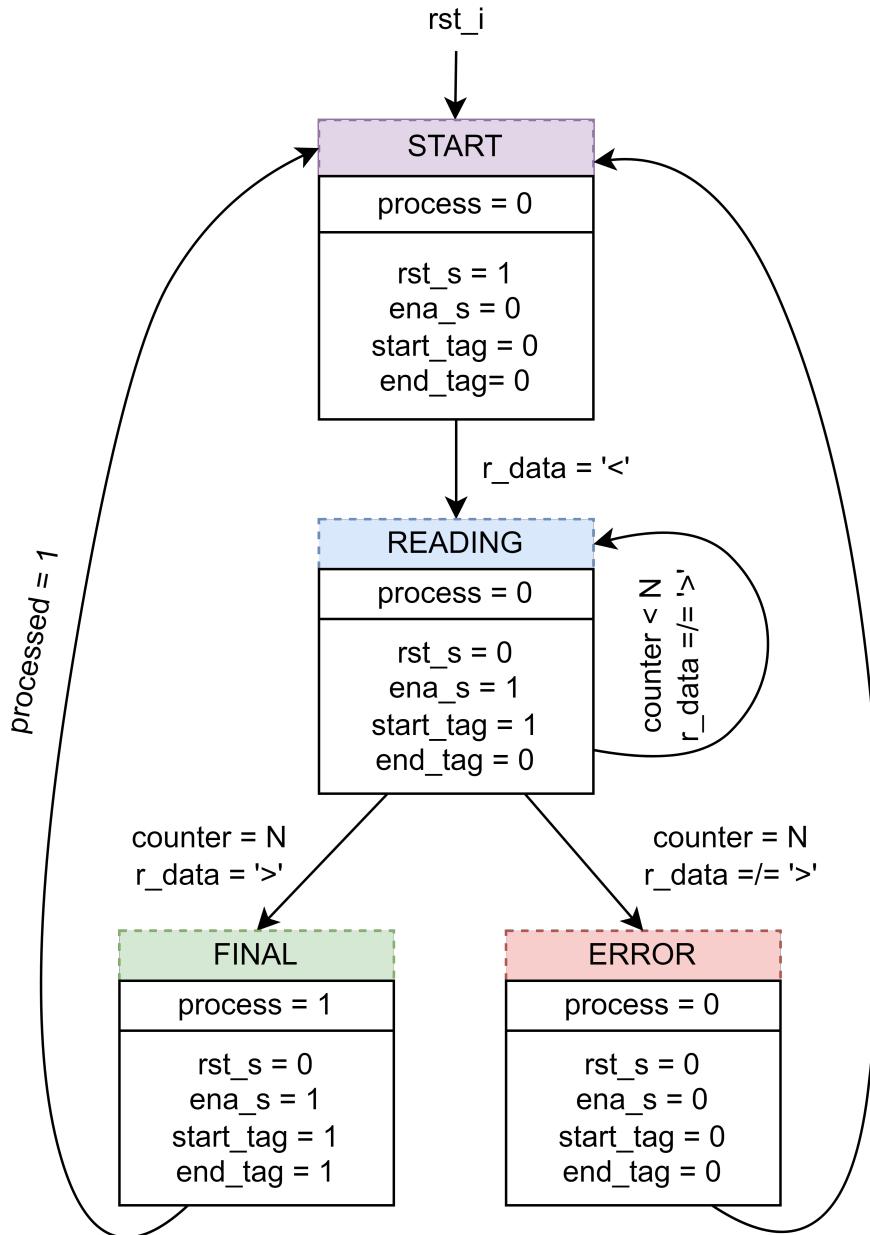


Figura 3.18: FSMD del módulo *Detector*.

En cada pulso de reloj (*clk.i*), el módulo UART, que se ilustra en la Figura 3.14, envía un carácter por medio de la señal *r_data* (8 bytes) y un pulso (*r_available*) para informar que un nuevo dato ha sido enviado. El pulso de reloj es utilizado principalmente en el módulo *Counter_0_to_N*, cuyo parámetro N ya ha sido calculado por el ACG previo a generar el código y es la cantidad de caracteres que serán enviados a continuación del carácter de inicio. El proceso de detección y validación de la trama recibida se describe el diagrama de estados de la Figura 3.19.

Figura 3.19: Diagrama de estados del módulo *Detector*.

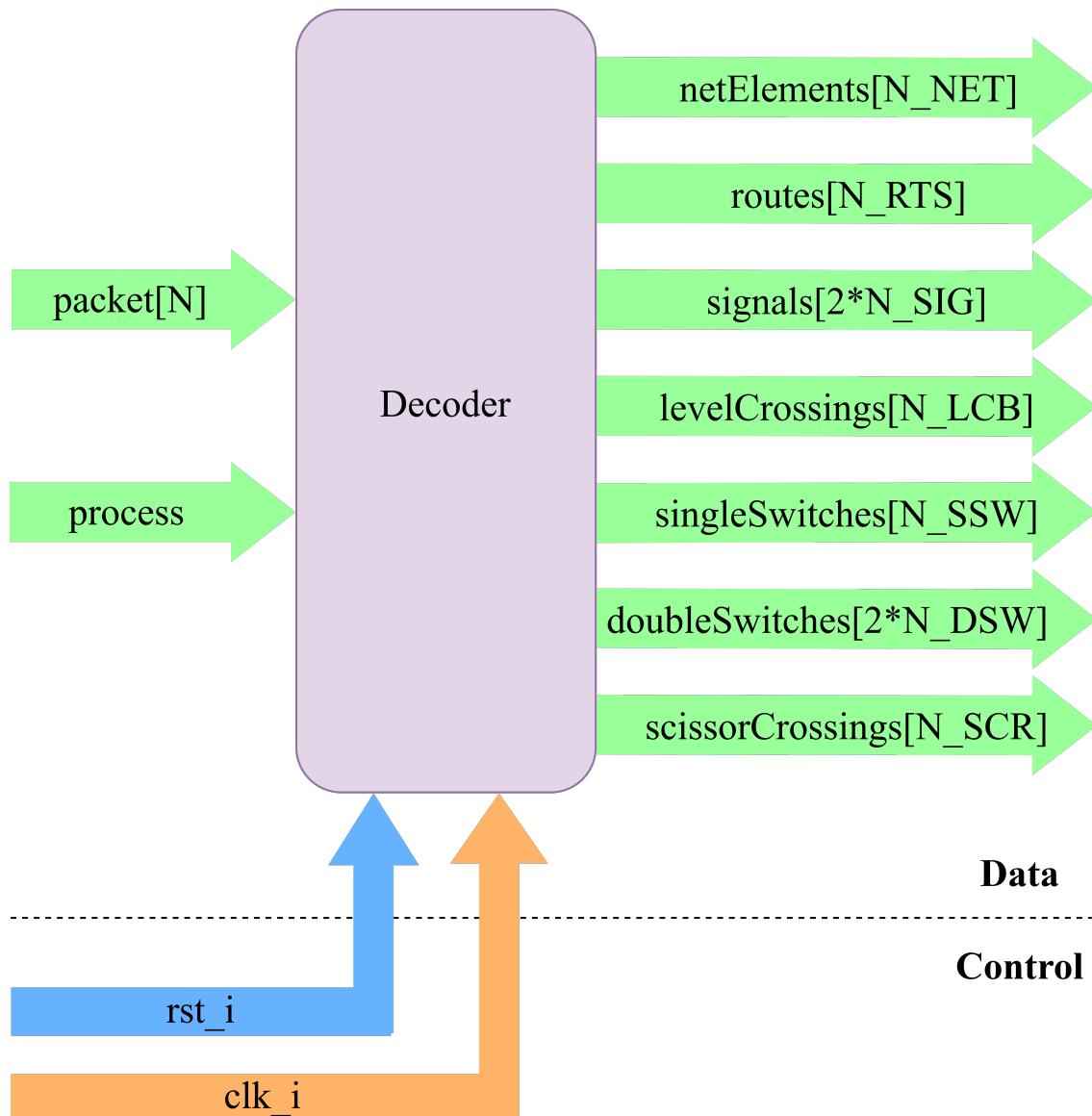
El módulo *Detector* inicia por detecto en el estado *start*, aguardando por el carácter de inicio de trama '`<`'. Al recibir el carácter de inicio de trama, el módulo transiciona al estado *reading*. En el estado *reading* se recibirán solamente los caracteres hexadecimales '0' a 'F'. Si al terminar de recibir N caracteres, el próximo carácter no es el de fin de trama '`>`' entonces se transiciona al estado *error*, se reinician las variables auxiliares, la trama se descarta y se vuelve al estado *start*.

Si el próximo carácter luego de leer N valores hexadecimales '0' a 'F' es el carácter de fin de trama '`>`', entonces el módulo transiciona al estado *final*, donde se da por válida la trama y se habilita su envío al módulo *decoder*, para volver al estado *start* a la espera de un nuevo carácter de inicio de trama, y se reinician 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 determinar rápidamente la fuente de eventuales errores.

3.2.3. Módulo Decoder

El módulo *Decoder* (ver Figura 3.14) es el encargado de demultiplexar la trama $packet[N]$ ya validada por el módulo *Detector*. El módulo *Decoder* recibe el vector de elementos booleanos $packet[N]$ y la señal *process* que indica cuando puede iniciar el proceso de demultiplexación. La salida serán todos los vectores de estado de los elementos ferroviarios. El diagrama de bloques de la máquina de estados finitos con camino de datos se muestra en la Figura 3.20. En este caso particular, no se cuenta con una máquina de estados, ya que la demultiplexación se realiza directamente.

Figura 3.20: FSMD del módulo *Decoder*.

La porción de $packet[N]$ correspondiente a cada vector será en función de la cantidad de elementos de cada tipo presentes en la locación. Esto ya fue calculado previamente por el ACG y explicado en la Sección 3.2.1 al definir el formato de la trama. Si la cantidad de un cierto elemento ferroviario es mayor que uno, el ACG implementará el estado de ese elemento con un vector hexadecimal del tamaño adecuado. Si solo existe un elemento ferroviario de ese tipo, el ACG implementará un escalar hexadecimal. Si no existiese ningún elemento ferroviario en la locación, el ACG no implementará ninguna de las funcionalidades relativas a dicho elemento, de esta forma se optimiza el uso de recursos en la FPGA.

3.2.4. Módulo Encoder

El módulo *Encoder* (ver Figura 3.14) es el encargado de multiplexar los vectores de estado provenientes del módulo *Network* y reconstruir una nueva trama, esta vez sin los valores de ocupación de vías por ser de sólo lectura. Esta nueva trama será $packet[M]$, junto con la señal *processed* para indicar al módulo *Printer* que la trama está completa y lista para ser transmitida. El diagrama de bloques de la máquina de estados finitos con camino de datos se muestra en la Figura 3.21.

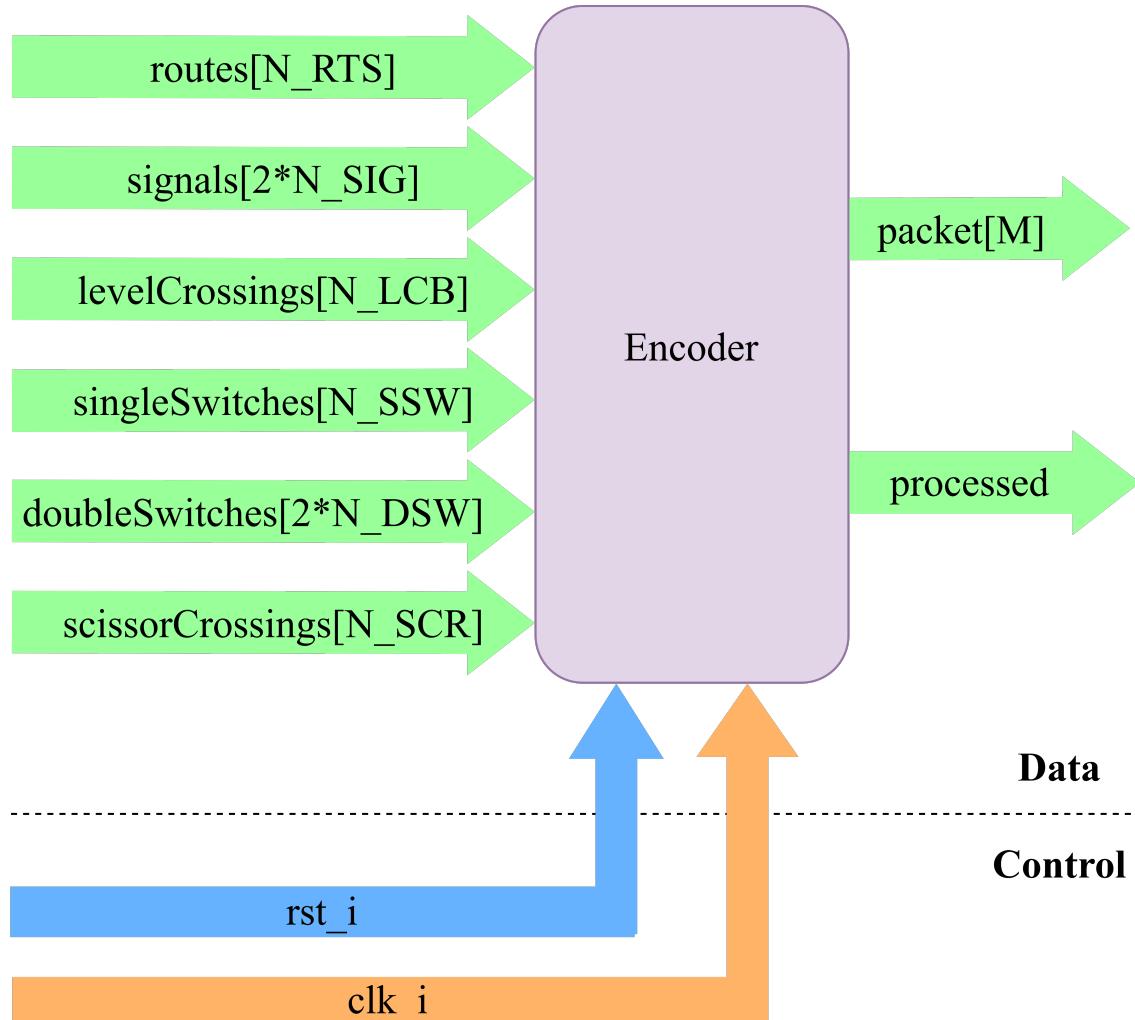


Figura 3.21: FSMD del módulo *Encoder*.

Al igual que la demultiplexación que realiza el módulo *Decoder*, la multiplexación se basa en la cantidad de elementos ferroviarios de cada tipo, calculados por el ACG al definir la trama tal cual fue explicado en la Sección 3.2.1. Los elementos inexistentes en la locación no serán tenidos en cuenta para formar la nueva trama, reduciendo el tamaño de la misma al mínimo.

3.2.5. Módulo Printer

El módulo Printer (ver Figura 3.14) realiza la conversión de cada elemento de un vector de M elementos hexadecimales ($packet[M]$, M elementos de 4 bits) en caracteres hexadecimales (1 byte). Cada elemento del vector es analizado en cada ciclo de reloj (clk_i) y demultiplexado, de manera tal de convertir un elemento por vez, para luego enviar el byte correspondiente al módulo UART para su posterior transmisión al exterior. El diagrama de bloques de la máquina de estados finitos con camino de datos se muestra en la Figura 3.22.

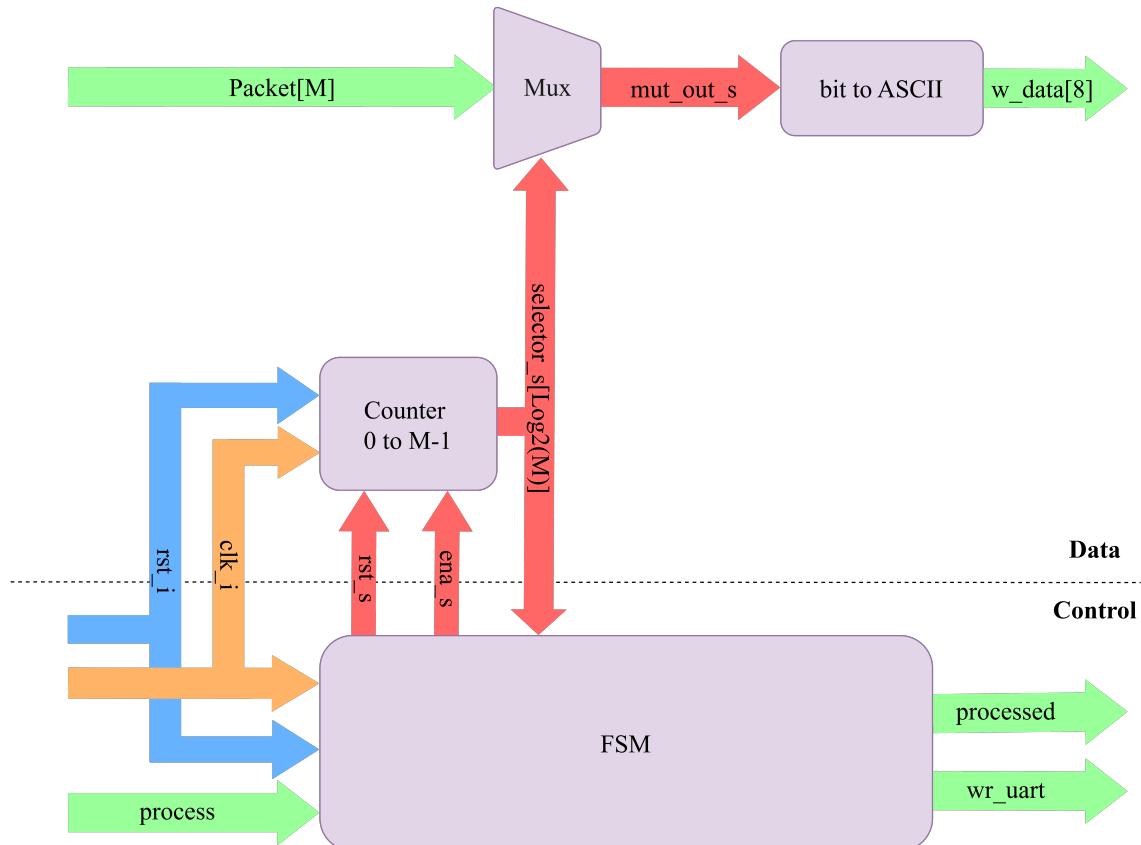
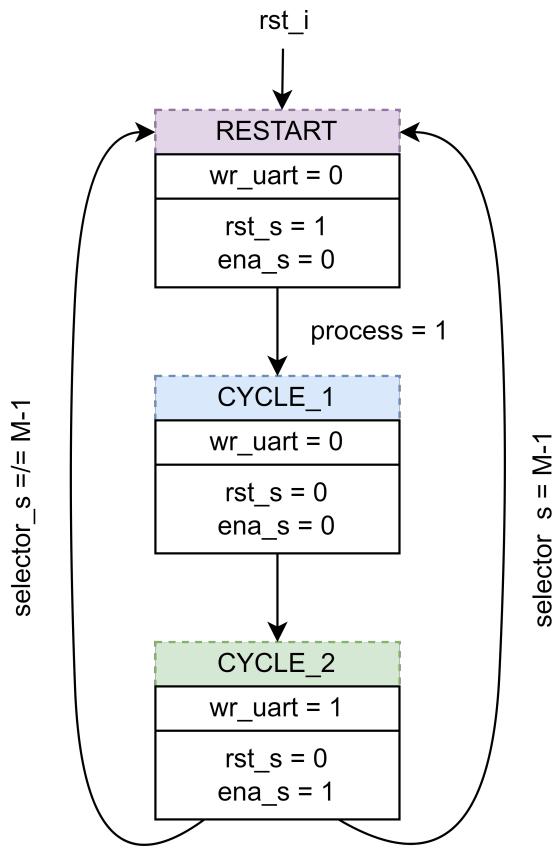


Figura 3.22: FSMD del módulo *Printer*.

En cada ciclo de reloj el módulo *Printer* demultiplexa el vector $packet[M]$ para obtener un elemento lógico que procesar, según el valor del contador vigente, que se incrementa en cada ciclo, hasta un máximo de $M-1$. Si el elemento $packet[i]$ es un valor hexadecimal, se enviará un byte equivalente en ASCII. Por ejemplo, se enviará un byte equivalente al 'A' ASCII si el elemento $packet[i]$ es una 'A' hexadecimal.

Cada dos ciclos de reloj el módulo *Printer* genera un pulso para habilitar el envío del último byte generado. Junto con el carácter se envía la señal wr_uart para indicarle a la UART que ese dato debe ser guardado en la FIFO de salida y la señal $processed$ para indicarle al módulo *Detector* que se pueden procesar nuevas tramas. El ciclo de procesamiento de la trama a transmitir se describe el diagrama de estados de la Figura 3.23.

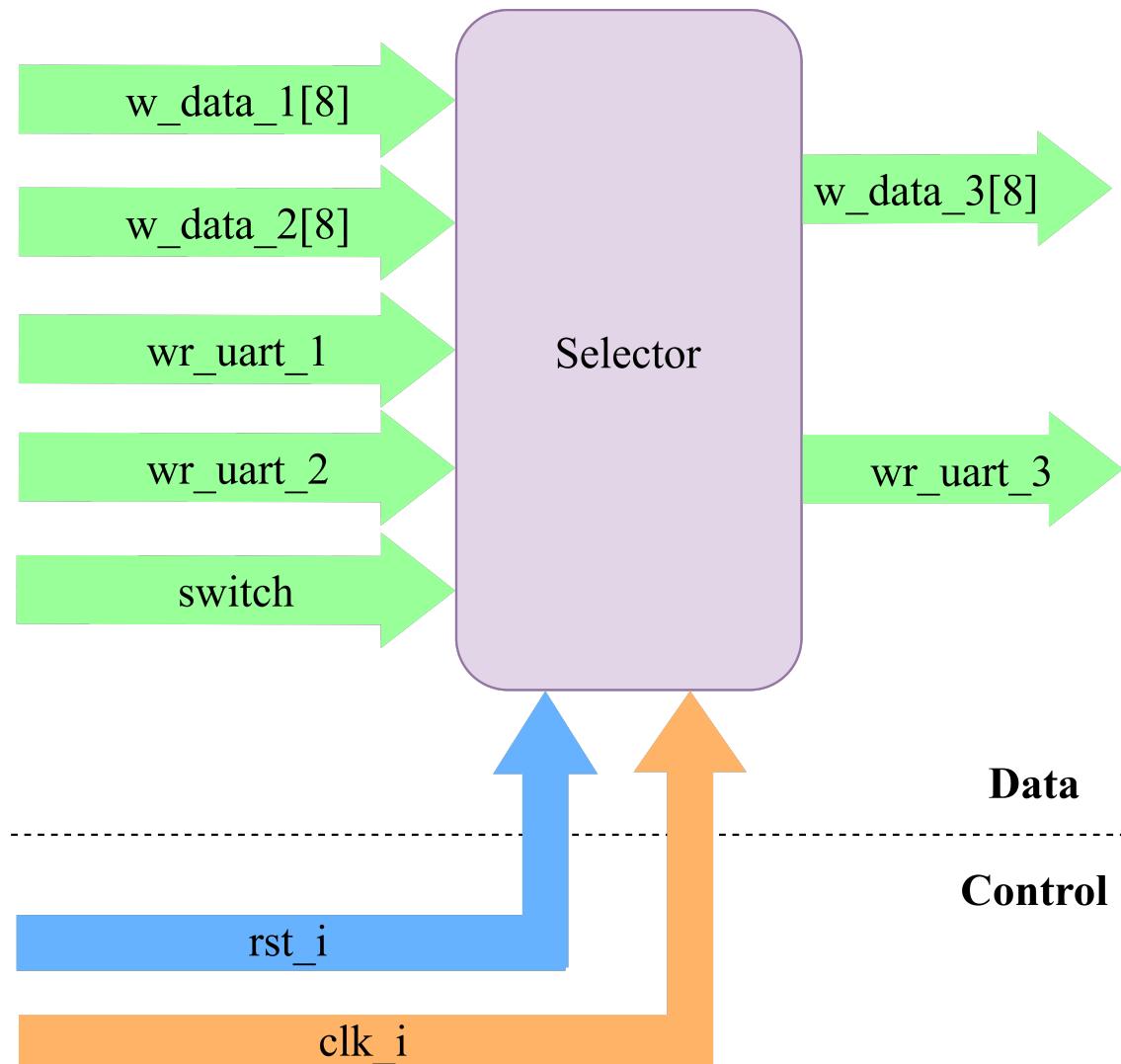
Figura 3.23: Diagrama de estados del módulo *Printer*.

El módulo *Printer* inicia por detecto en el estado *restart*, a la espera de recibir la señal *process* del módulo *Encoder*. Se tienen dos estados (*cycle_1* y *cycle_2*) para generar el pulso de reloj necesario para mantener sincronizadas las tramas. Cuando el contador haya recorrido los M elementos de *packet[M]*, el módulo vuelve al estado *restart*, para esperar una nueva señal *process* para volver a procesar una nueva trama de datos.

Si la trama recibida es incorrecta, o si ya fue impresa, entonces la señal *process* será '0' y el modulo *Printer* dejará de enviar datos a la UART. Si la señal *process* mantiene un estado lógico positivo, el proceso de impresión continuará hasta que la UART indique que no pueda recibir mas datos o que alguna etapa previa informe de algún error en el proceso.

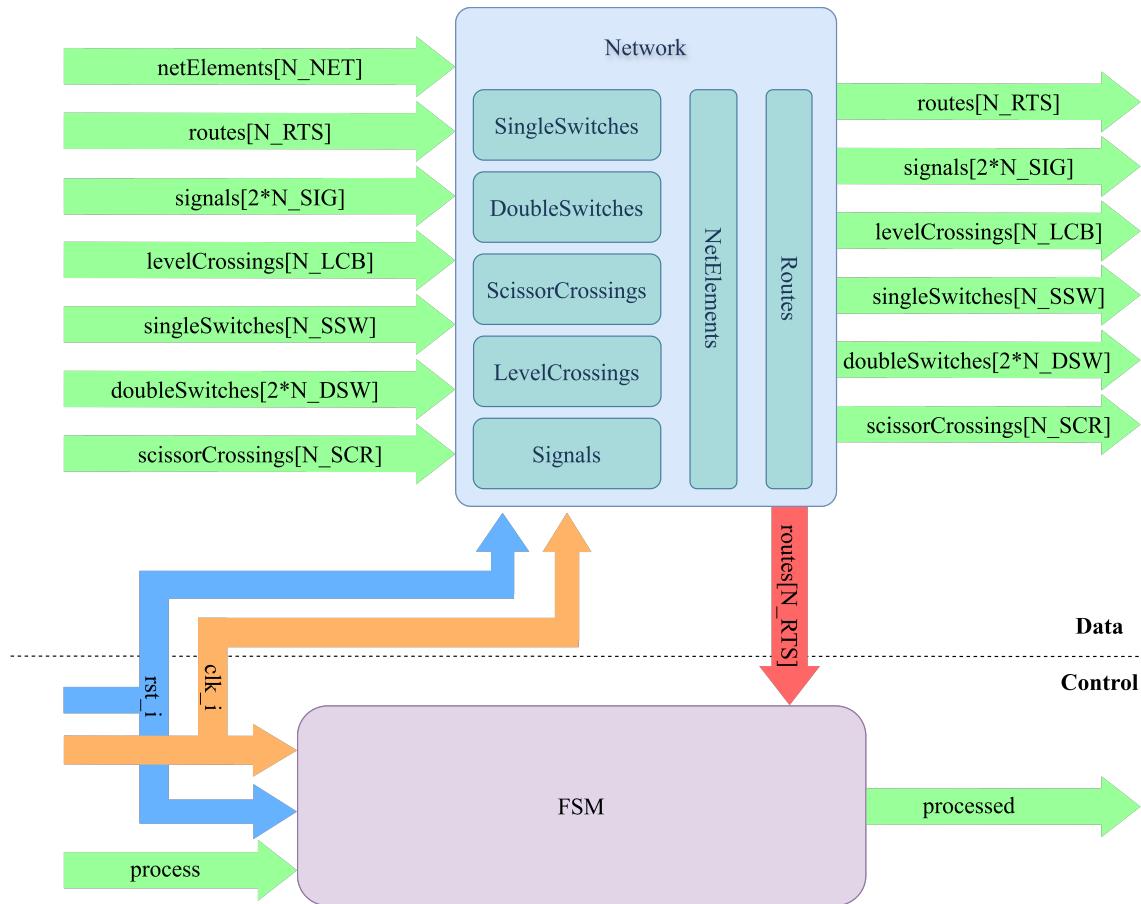
3.2.6. Módulo Selector

Se añadió el módulo *Selector* (ver Figura 3.14) para poder facilitar el testeo de la comunicación serial al permitir anular la totalidad del sistema de enclavamiento. De esta manera, es posible validar la lectura, detección y escritura de tramas en bucle en forma independiente al sistema de enclavamiento. Esta funcionalidad es habilitada cambiando la posición de un switch físico de la FPGA y se desactiva invirtiendo su posición. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.24.

Figura 3.24: FSMD del módulo *Selector*.

3.2.7. Modulo Network

El módulo *Network* (ver Figura 3.14) es el encargado de instanciar todos los elementos ferroviarios presentes en la red de grafos generada por el RNA e interconectarlos como el RNA indica. Es el módulo que mas recursos de la FPGA utiliza y donde más se hace énfasis en el uso óptimo de los recursos. Para lograr esto, se prioriza la máxima descentralización posible de los módulos instanciados internamente, de forma tal que cada uno de ellos solo dependa de la mínima cantidad necesaria de módulos, evitando conexiones innecesarias. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.25.

Figura 3.25: FSMD del módulo *Network*.

Los módulos de *NetElements*, *Routes* y *Signals* son obligatorios ya que se encuentran presentes en la mínima red ferroviaria aceptada por el RNA. En cambio, los módulos de *SingleSwitches*, *DoubleSwitches*, *ScissorCrossings* y *LevelCrossings* son opcionales y dependen de que existan en la locación. Es decir, si un elemento ferroviario no existe, no solamente no serán instanciado por el ACG, sino que tampoco se generarán archivos que definan sus módulos, señales auxiliares que los interconecten a otros módulos ni tampoco entradas o salidas en otros módulos, destinados a este elemento. De esta manera, se reduce la complejidad del sistema enormemente al implementar solamente los puertos, señales, módulos y conexiones que serán utilizados por el sistema de enclavamiento.

Para garantizar la descentralización de la red, a todos los elementos ferroviarios con entidad física, es decir, todos los mencionados (vías, pasos a nivel, cambios de vías, señales), excepto las rutas que son entidades abstractas, se les implementa la propiedad de enclavamiento. El enclavamiento de un elemento puede presentar tres estados, tal como se describe en la Tabla 3.1.

Tabla 3.1: Estados de enclavamiento de cada elemento ferroviario.

Liberado	Reservado	Enclavado
Solicitado por al menos una ruta u opera de forma automática.	Operado solamente por la ruta que lo reservó.	Depende de sí mismo y de la ruta que lo enclavó.

Todo elemento se encuentra por defecto en el estado liberado y puede ser solicitado por cualquier ruta. Tan pronto una ruta envía el comando de reserva al elemento, pasa a estar en estado reservado y ninguna otra ruta podrá hacer uso de este elemento. Finalmente, si la ruta ha podido reservar todos los elementos necesarios para cumplir las condiciones para ser habilitada, enviará la señal de enclavamiento a todos los elementos involucrados, lo que evita que puedan ser comandados por otras rutas. Esto incluye tanto a la ocupación de vías (*NetElements*) como a las señales (*Signals*), pasando por todos los cambios de vías (*SingleSwitches*, *DoubleSwitches* y *ScissorCrossings*) y barreras de pasos a nivel (*LevelCrossings*). De esta manera el ACG se asegura que cada elemento sea controlado por una única ruta por vez o de forma automática si ninguna ruta demanda su uso.

Para generalizar y modelizar el comportamiento dinámico de los elementos ferroviarios se utilizaron redes de Petri [95, 221-225, 246]. Una red de Petri es la representación gráfica de un modelo matemático de autómatas concurrente. Las mismas están formadas por lugares (los estados), las transiciones que indican las condiciones para pasar de un lugar a otro, y los arcos que conectan lugares y transiciones. Las redes de Petri pueden poseer uno o más tokens que representan en qué estado se encuentra el sistema. Algunas redes de Petri admiten más de un token por red, o incluso más de un token por lugar. En el caso del sistema de enclavamiento se definieron solamente redes de Petri con un único token por red, siendo que el sistema solo puede adoptar un estado en cada momento. En la Figura 3.26 se visualiza el modelo matemático del enclavamiento de un elemento ferroviario genérico.

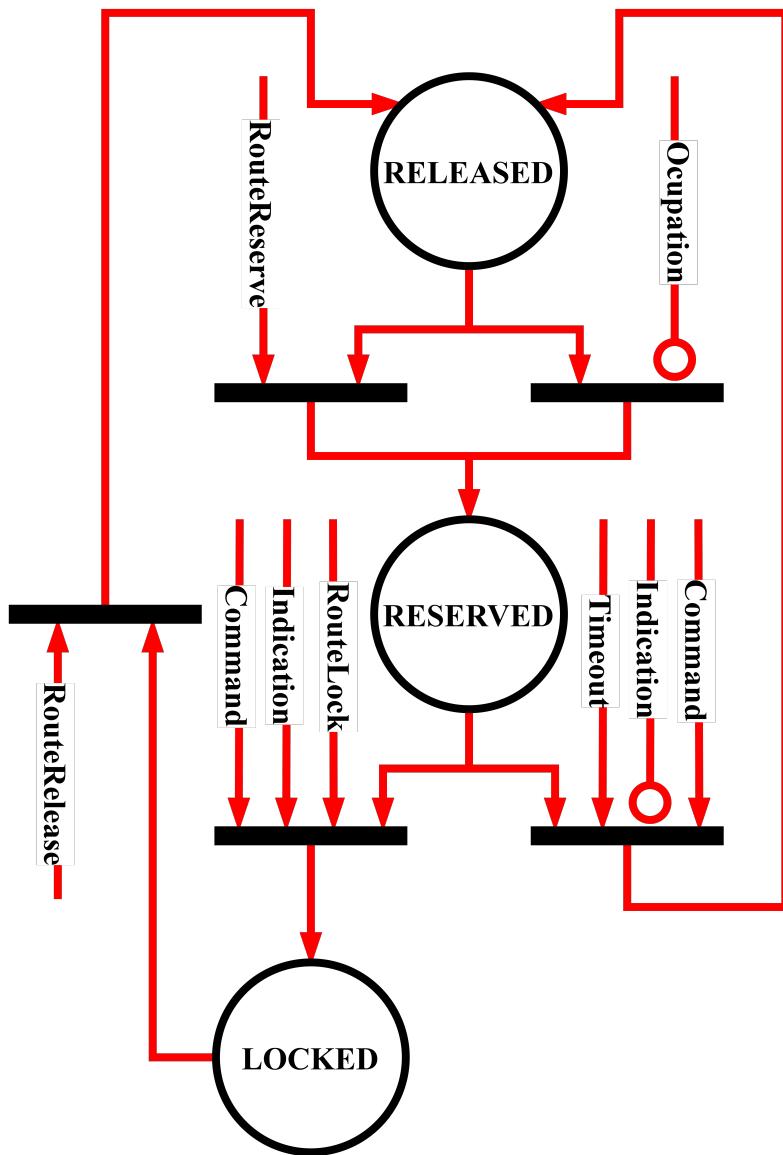


Figura 3.26: Red de Petri del modelo dinámico de enclavamiento de elementos ferroviarios genéricos.

En la Figura 3.26 se observa que la transición de un elemento ferroviario del estado liberado a reservado no solo puede darse por el pedido expreso de una ruta que lo solicite, sino también de facto al ocuparse los netElements cercanos al elemento, lo cual inicia lo que se denomina 'bloqueo por ocupación' (ver Sección 3.1.1). Esto protege al sistema al impedir que otras rutas soliciten un cambio de estado de un elemento que está siendo usado por una formación que no ha solicitado ningún permiso, lo que reduce el riesgo de accidentes por descarrilamiento o colisión.

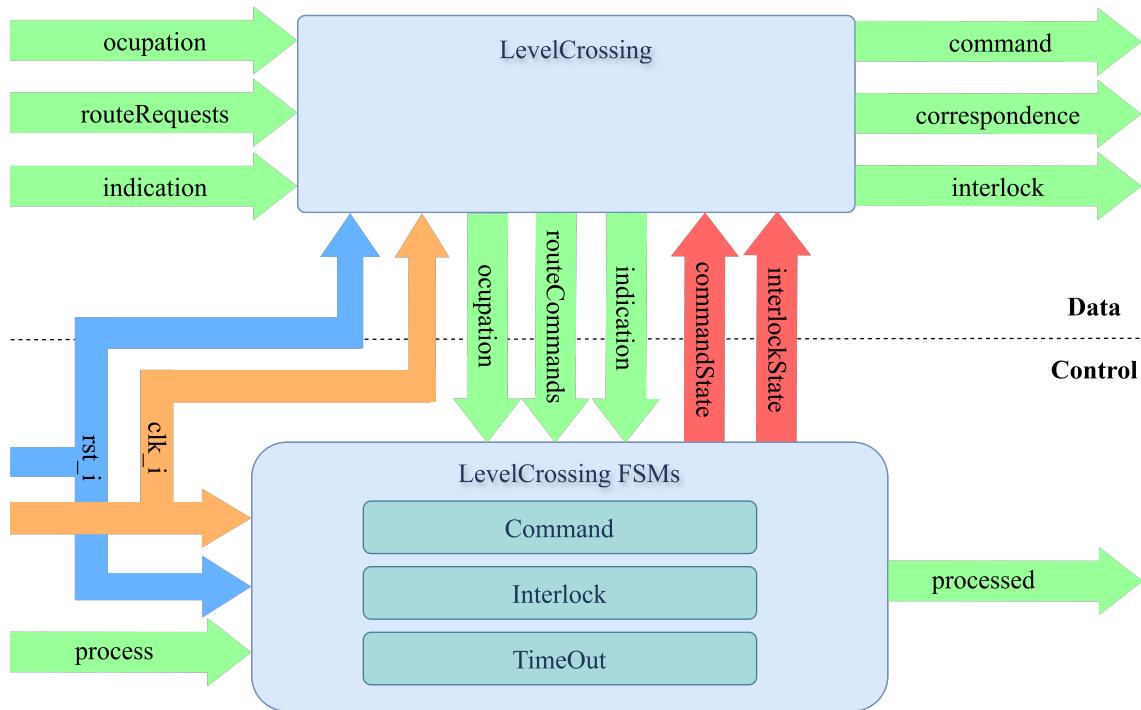
La red de Petri también deja explícito la inclusión de un timeout para las reservas, con un valor por defecto de 7 segundos, configurable por el usuario. Este valor fue seleccionado por ser el que se utiliza actualmente en la mayoría de la industria ferroviaria [247]. De no cumplirse que el comando y la indicación coincidan dentro del tiempo estipulado, el elemento es liberado, siempre que no se encuentren formaciones cercanas. Las transición de reserva a enclavamiento requiere que

tanto el comando como la indicación coincidan y que la ruta solicite el enclavamiento al haber confirmado que todas las condiciones de ruta se han cumplido y es inminente su habilitación. El elemento ferroviario deja de estar enclavado cuando la ruta lo solicita expresamente, momento en el cuál pasa a estar liberado para que otras rutas puedan usarlo.

A diferencia de los módulos explicados en secciones anteriores, donde cada módulo era instanciado una vez, los módulos que modelan elementos ferroviarios, con entidad física (cómo un cambio de vías simple) o abstracta (cómo una ruta ferroviaria), son instanciados tantas veces como unidades de éstos existan. Es por eso que, en las siguientes secciones, hablaremos de módulos genéricos a la hora de describir cada módulo. Estos módulos genéricos presentarán todas las entradas, salidas y funcionalidades que pueden tener, en caso de ser necesarias para la implementación del sistema de enclavamientos. Es trabajo del ACG determinar cuales de ellas deben ser implementadas y cuales no. Por ejemplo, si una ruta A requiere que una barrera X se encuentre baja, pero no tiene requerimientos respecto a las posiciones de los cambios porque la ruta A no los utiliza, entonces el ACG instanciará el módulo de la ruta A considerando en las entradas y salidas los estados de la barrera X y los comandos para operarla, interconectando el módulo de ruta A con el módulo de la barrera ferroviaria X. En cambio, el ACG no implementará ningún puerto destinado a los cambios de vías ni lo considerará en las funcionalidades del módulo de ruta A. De igual manera, el ACG conectará los estados de ocupación de vías correspondientes solamente a los módulos de los elementos ferroviarios que los necesitan y no la totalidad del vector de datos.

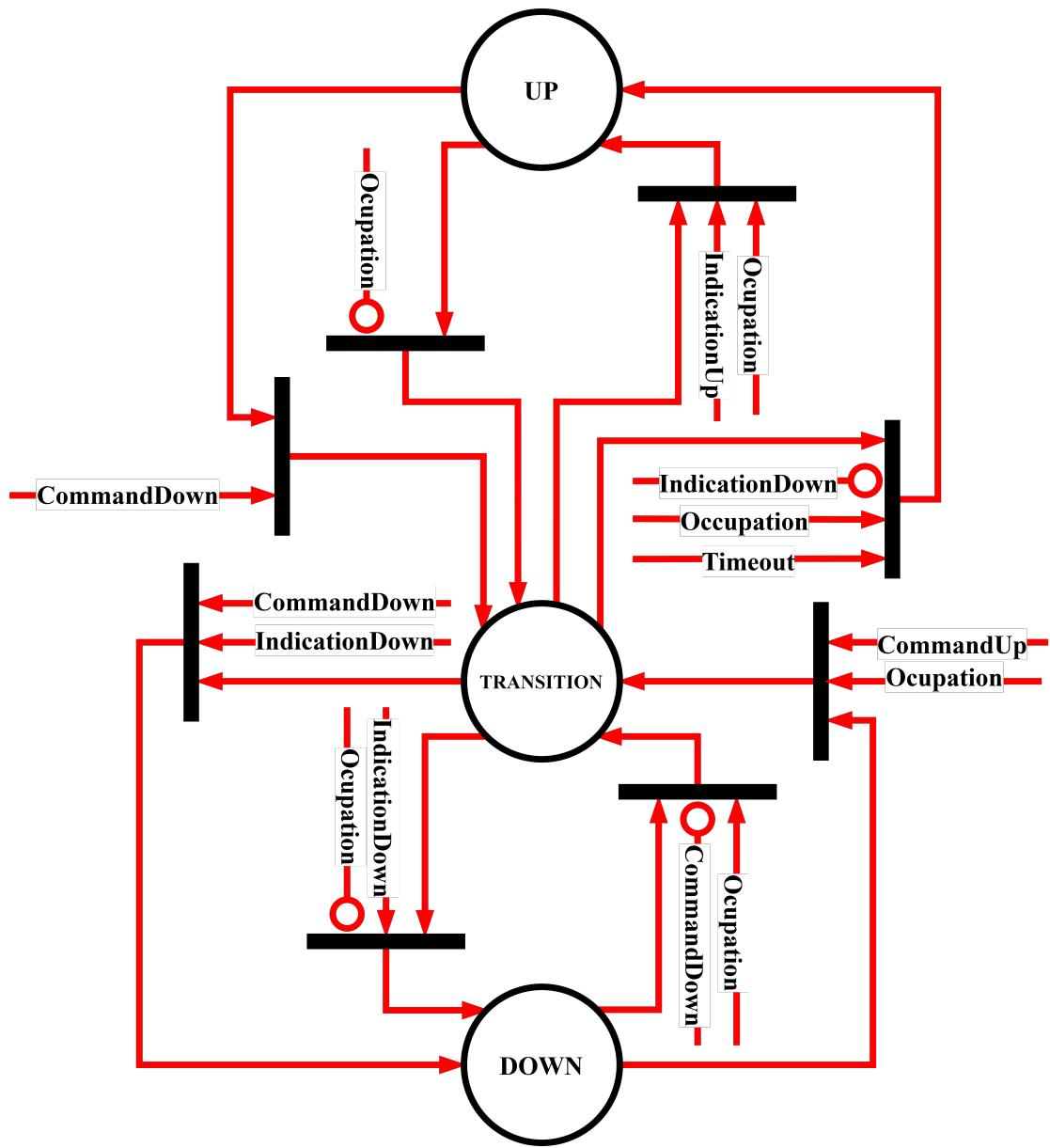
3.2.8. Módulo genérico de las barreras ferroviarias

El módulo *LevelCrossing* (ver Figura 3.14) es el encargado de implementar el funcionamiento de las barreras ferroviarias en los pasos a nivel. El ACG utiliza la información otorgada por el RNA para determinar cuál es el *netElement* donde se sitúa el paso a nivel y cuales son los *netElements* mas próximos, para que puedan reportar su estado al módulo *LevelCrossing*. Además, el ACG implementa todas las conexiones para cada una de las rutas en las cuales el paso a nivel sea condición necesaria para su habilitación. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.27.

Figura 3.27: FSMD del módulo genérico de *LevelCrossing*.

Tal como se explicó en la Sección 2.1.9, las barreras ferroviarias también poseen comando, indicación y correspondencia: comando para controlar la barrera, indicación para reportar su estado actual al módulo *LevelCrossing* y correspondencia para declarar su estado final a la ruta que lo solicite. Además, el ACG implementa todas las conexiones necesarias a todas las rutas que tengan a la barrera en cuestión como condición necesaria de habilitación. Finalmente, el módulo *LevelCrossing* tiene una salida que define su estado de enclavamiento frente a otras rutas que lo requieran.

El comportamiento de una barrera de paso a nivel genérico se define en la red de Petri de la Figura 3.28. Una barrera se mantendrá en estado alto a menos que una ruta solicite que el brazo de barrera descienda o si los *netElements* cercanos se encuentran ocupados. Durante la transición se espera la confirmación de que la barrera ha descendido mientras se mantiene la ocupación para terminar el movimiento en el estado bajo, estado en el cual solo podrá salir si la ocupación termina y ninguna ruta solicita que la barrera se mantenga baja.

Figura 3.28: Red de Petri del modelo dinámico de *LevelCrossing*.

Durante el estado de transición la barrera tendrá 7 segundos para llegar al estado bajo. Este parámetro es configurable, pero se eligió por defecto los 7 segundos que se utilizan en las barreras ferroviarias por diseño. Al cumplirse este tiempo, si la indicación y el comando no coinciden y los *netElements* cercanos se encuentran libres, la barrera volverá al estado alto. Esto impide que el sistema habilite rutas cuando los actuadores de algún elemento se atascan o su indicación es diferente al comando que se ha enviado. La confirmación de la indicación otorga un grado de seguridad mayor al no asumir que un comando enviado implica automáticamente que al actuador se le impone dicho estado.

3.2.9. Módulo genérico de los cambios de vías simples

El módulo *SingleSwitches* (ver Figura 3.14) es el encargado de implementar el funcionamiento de los cambios de vías simples en la red ferroviaria. El ACG utiliza la información otorgada por el RNA para determinar cuales son los *netElements* conectados mediante el cambio de vías, los *netElements* más próximos. Esta información se utiliza para implementar las entradas del módulo *SingleSwitches* donde se reporta el estado de ocupación de los *netElements* (*occupation*). Además, se implementan las conexiones de indicación, comando y correspondencia para poder leer, escribir y reportar el estado del cambio de vías, respectivamente. Finalmente, el ACG implementa las conexiones a todas las rutas que buscarán controlar el cambio de vías y todo los mecanismos para obedecer solo a una de ellas, de cumplirse las condiciones. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.29.

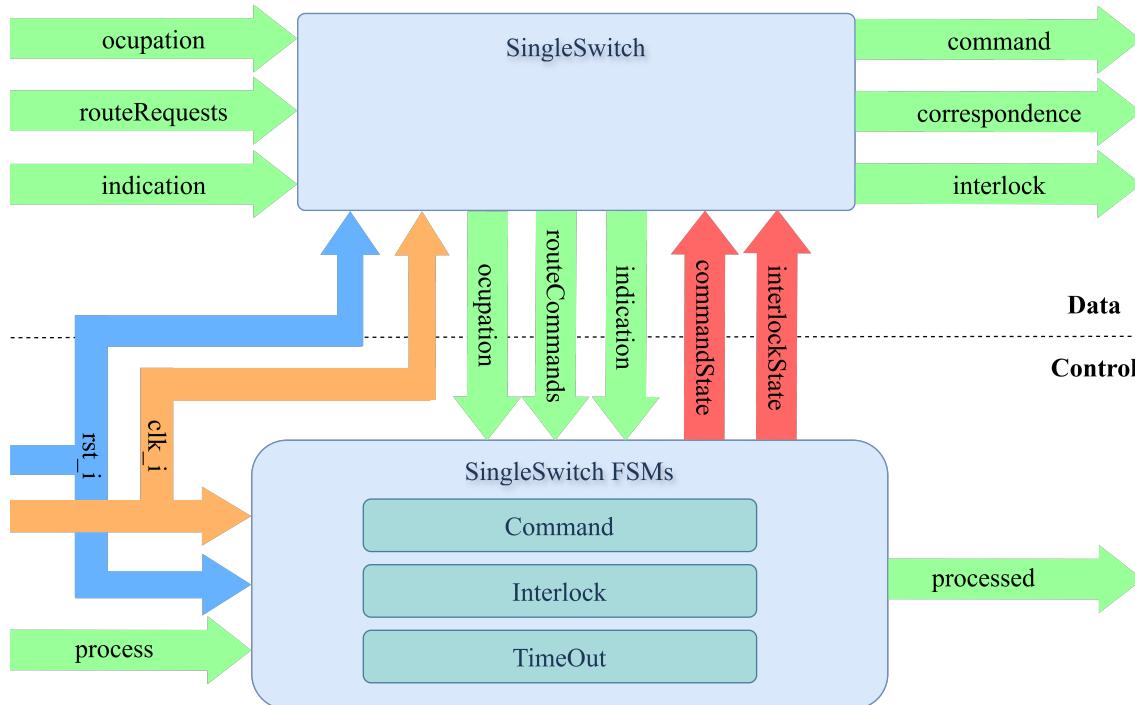
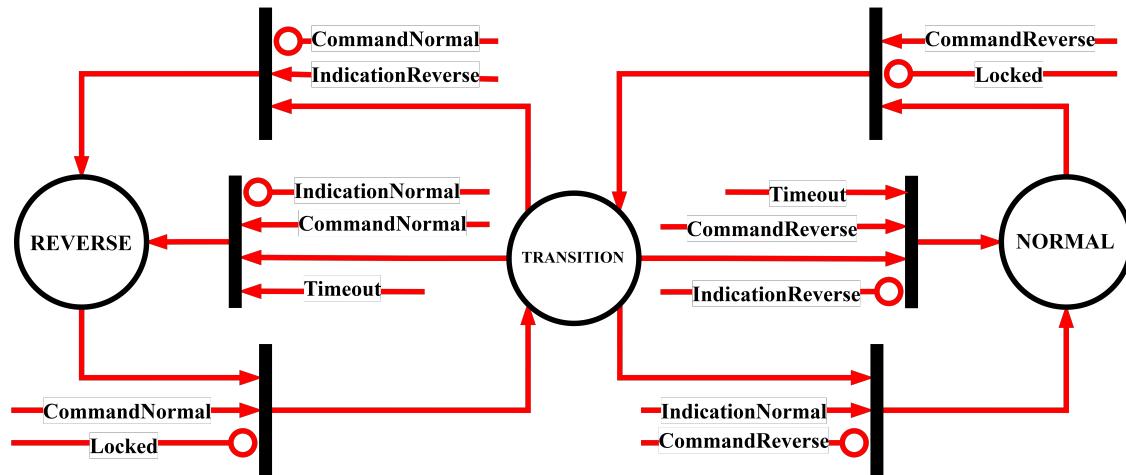


Figura 3.29: FSMD del módulo genérico de *SingleSwitches*.

Como fue explicado en la Sección 2.1.9, un cambio de vías simple puede adoptar dos posiciones: normal y reversa. Adicionalmente, es necesario contemplar que mientras la indicación no reporte una posición definida, normal o reversa, se deberá asumir que el cambio de vías se encuentra en transición de un estado al otro. Dicha transición debe ser corta, ya que de no completarse el movimiento puede ser tanto que el cambio de vías se encuentre atascado como que el comando o la indicación se encuentren desconectados del actuador. Esta funcionalidad, junto con el comportamiento del cambio de vías genérico se define en la red de Petri de la Figura 3.30.

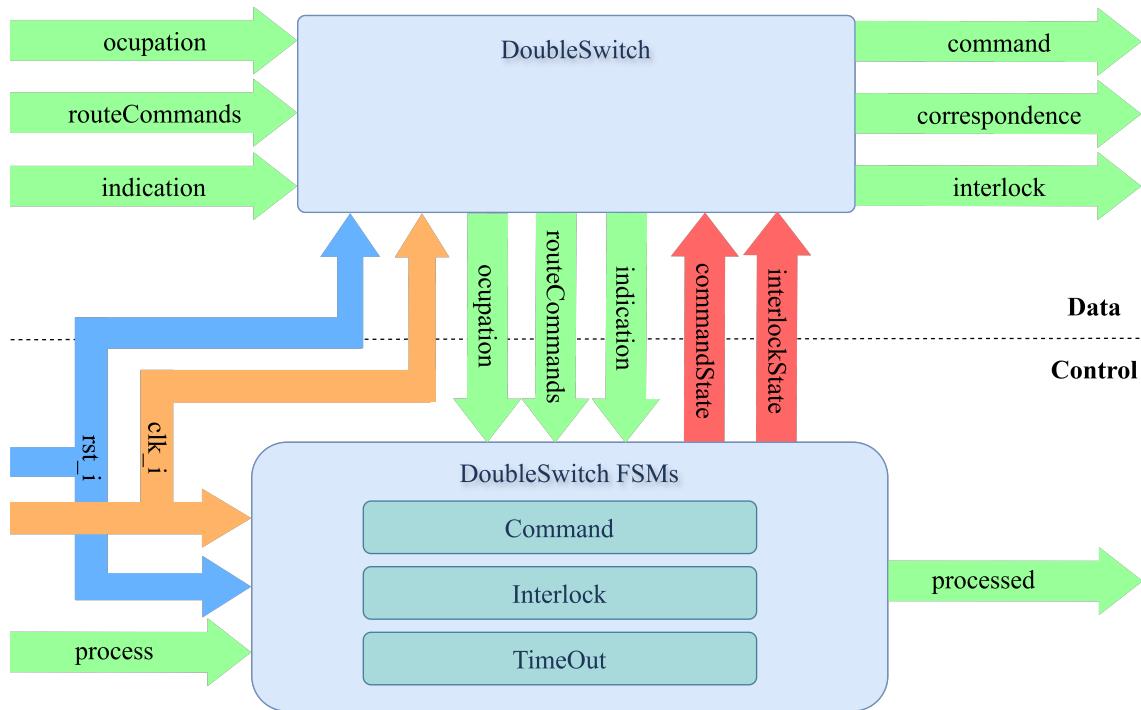
Figura 3.30: Red de Petri del modelo dinámico de *SingleSwitches*.

Un cambio de vías en estado reversa iniciará su transición si alguna de las rutas envía un comando para modificar la posición a normal, siempre que el cambio de vías no se encuentre enclavado por otra ruta que lo esté utilizando. Durante la transición, salvo que ocurra un timeout, el cambio de vías pasará al estado normal si se confirma la indicación normal y no el comando normal no fue cancelado por la ruta. Análogamente, el cambio de vías puede pasar del estado normal al reversa mediante la secuencia opuesta. En ambos casos, se deben cumplir las mismas condiciones de no estar enclavado antes de mover el cambio de vías, confirmación de indicación y comando, y realizar todo el proceso dentro de un tiempo determinado.

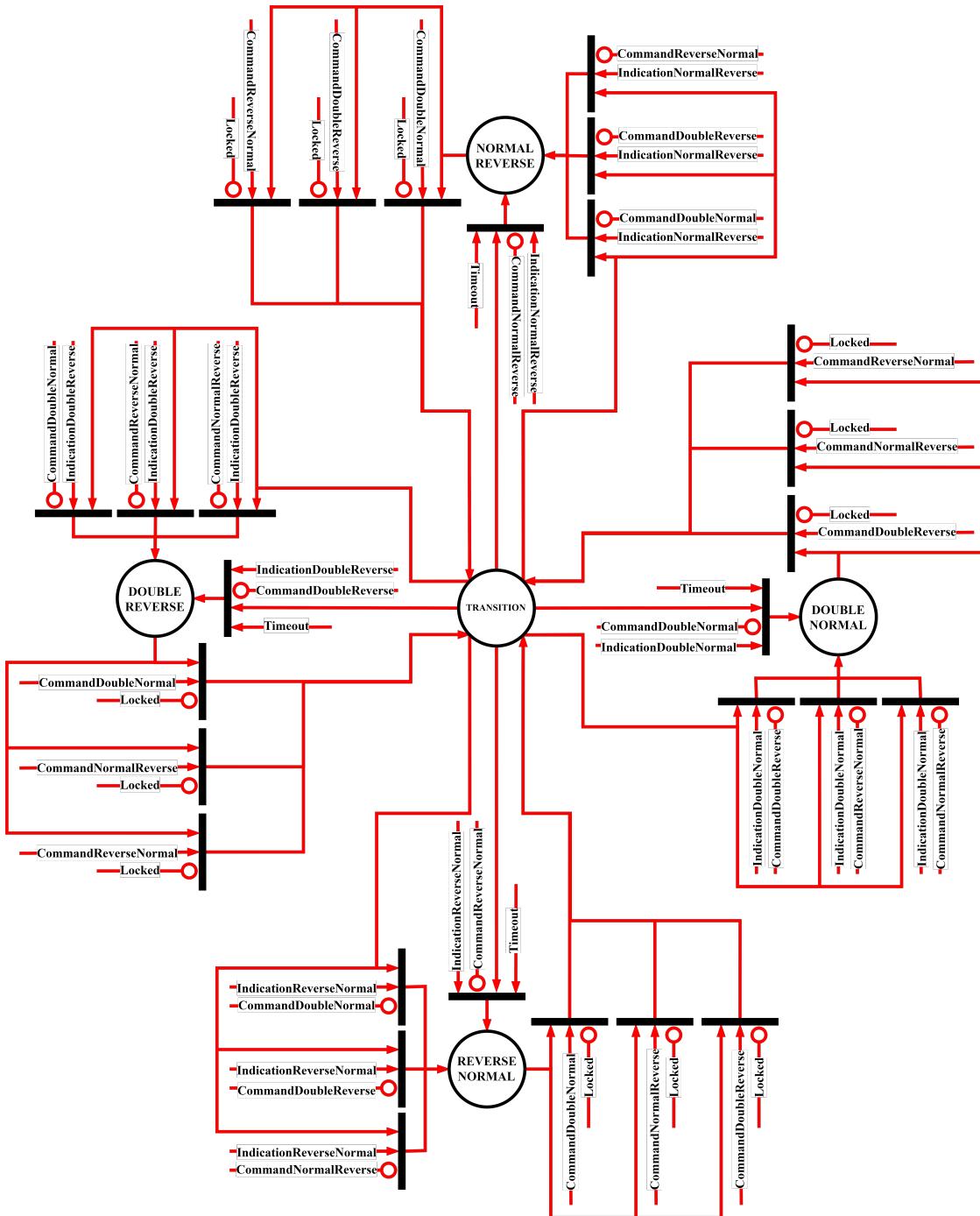
En el caso del timeout, se considera que si el cambio de vías no reporta una posición final en menos de un tiempo determinado, entonces el cambio de vías debe volver al estado anterior, de ser posible. De lograrse este o no, el cambio de vías igualmente quedará anulado y no podrá ser utilizado por la ruta que lo demanda, prohibiendo la habilitación de dicha ruta.

3.2.10. Módulo genérico de los cambios de vías dobles

El módulo *DoubleSwitches* (ver Figura 3.14) es el encargado de implementar el funcionamiento de los cambios de vías dobles en la red ferroviaria. Al igual que para el módulo *SingleSwitches*, el ACG utiliza la información otorgada por el RNA para implementar las mismas entradas y salidas en el módulo *DoubleSwitches*. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.31.

Figura 3.31: FSMD del módulo genérico de *DoubleSwitches*.

Aunque en una primera inspección las Figuras 3.29 y 3.31 puedan parecer muy similares, la diferencia entre ambos módulos radica en el tipo de datos utilizados en la indicación, comando y correspondencia, ya que los cambios de vías dobles tienen cuatro estados y no dos como los cambios de vías simples, tal se explicó en la Sección *SingleSwitches*. Un cambio de vías dobles puede adoptar las posiciones doble normal, doble reverso, normal reverso y reverso normal. Por lo tanto, señales que pueden adoptar dos estados no son suficientes y deben definirse nuevos tipos de datos que utilizan el doble de tamaño para poder definir cuatro estados. Al tener esto en cuenta, el comportamiento de los cambios de vías dobles es mucho más complejo, tal como se define en la red de Petri de la Figura 3.32.

Figura 3.32: Red de Petri del modelo dinámico de *DoubleSwitches*.

La complejidad de la red de Petri que define el comportamiento del módulo *DoubleSwitches* es mayor que los casos expuestos anteriormente al atenderse una mayor cantidad de posibles transiciones y estados. No obstante, el principio es el mismo que el explicado para el módulo *SingleSwitches*. Cada estado estable pasa al estado de transición si y solo si se una ruta envía un comando para que

el cambio de vías se desplace a una posición diferente y el cambio de vías no se encuentra enclavado por otra ruta. Una vez en transición, se dispone de una cantidad de tiempo finita (configurable) para completar el movimiento. De no lograrlo, el cambio de vías dobles deberá volver a su posición original, de ser posible. De cumplirse las condiciones adecuadas donde el comando y la indicación coinciden, el cambio de vías dobles alcanza un nuevo estado estable, reportando su posición actual a la ruta que lo solicitó, mediante su correspondencia.

3.2.11. Módulo genérico de los cambios en tijeras

El módulo *ScissorCrossings* (ver Figura 3.14) es el encargado de implementar el funcionamiento de los cambios de vías en tijeras en la red ferroviaria. Su implementación es similar a la del módulo *SingleSwitches*, con el ACG utilizando la información otorgada por el RNA para implementar las entradas y salidas necesarias para habilitar el movimiento del cambio de vías solo en situaciones seguras y confirmando su correspondencia mediante la comparación del comando y la indicación. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.33.

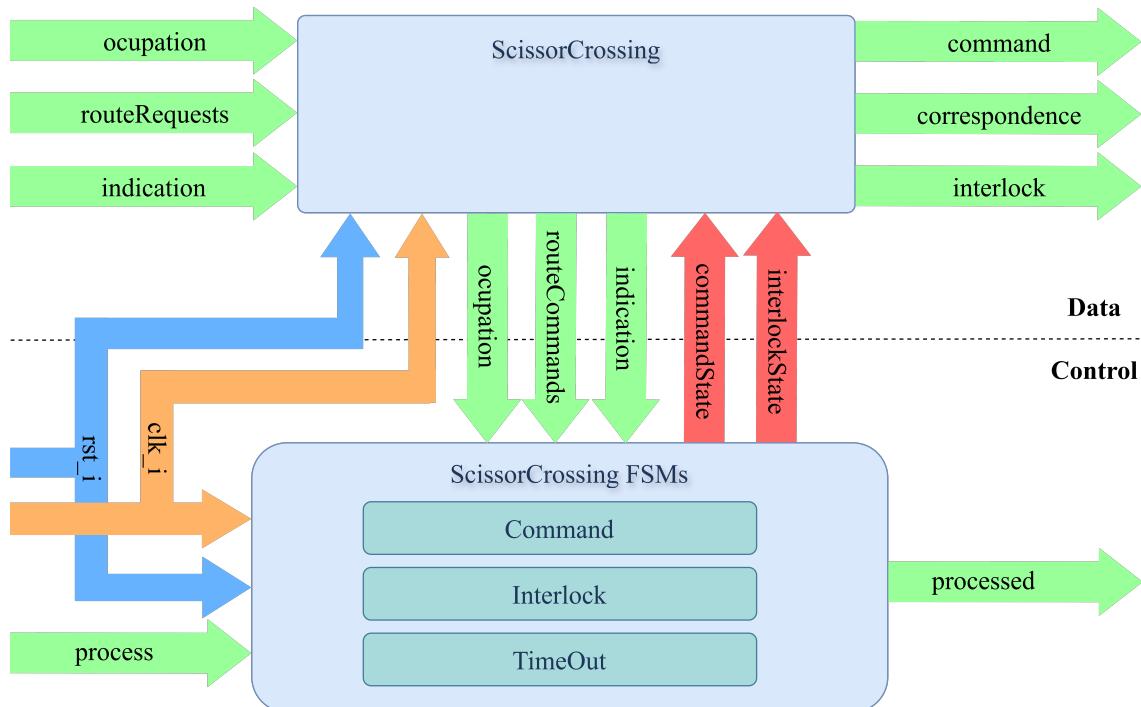
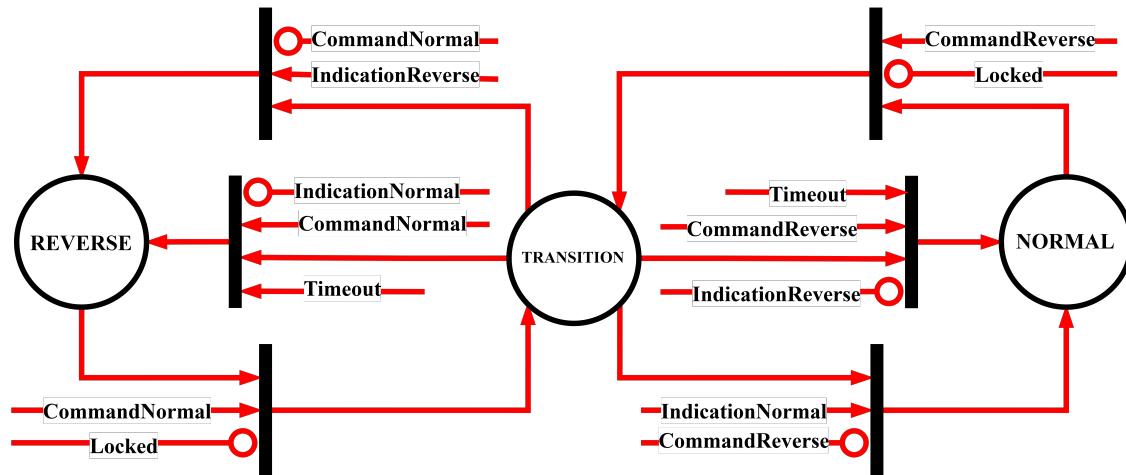


Figura 3.33: FSMD del módulo genérico de *ScissorCrossings*.

Tal se explicó en la Sección *SingleSwitches*, los cambios de vías en tijeras tienen dos posiciones estables: normal y reversa. A diferencia de los cambios de vías simples, donde la posición normal es la posición de mayor prioridad, ya que permite la circulación por vía principal; los posiciones en los cambios de vías en tijeras tienen igual prioridad, ya que ambas posiciones permiten la circulación por vías de igual categoría, pudiendo ser ambas principales. Por lo tanto, aun cuando por defecto se mantiene la funcionalidad de que un cambio de vías retorne a su posición original luego de llegar al timeout sin poder concretar el movimiento, esta función puede desactivarse y dejar el cambio en una posición intermedia luego de un timeout. El comportamiento de los cambios de vías dobles es mucho más complejo, tal como se define en la red de Petri de la Figura 3.34.

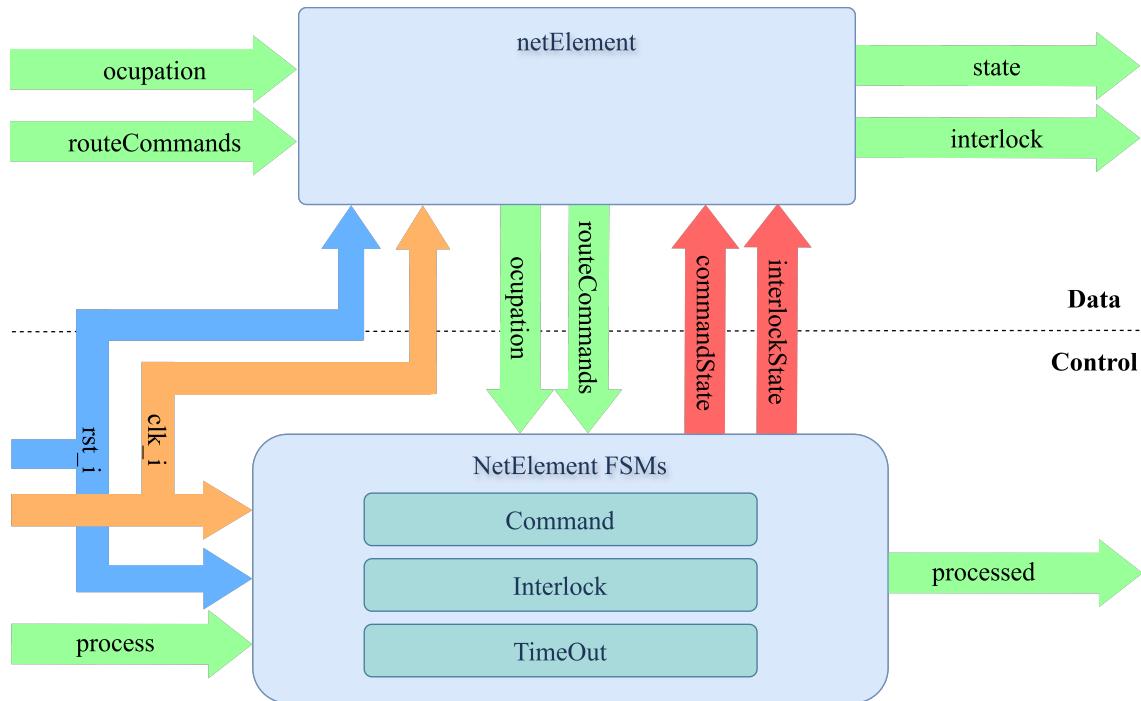
Figura 3.34: Red de Petri del modelo dinámico de *ScissorCrossings*.

En definitiva, los cambios de vías presentan diferencias en sus modelos dinámicos de comportamiento y en el tamaño de los puertos de comando, indicación y correspondencia, pero a grandes rasgos presentan muchas similitudes que son aprovechadas por el ACG. Entre las ventajas encontradas tenemos el uso de una plantilla común a la hora de definir los módulos y sus máquinas de estados, lo que agiliza el proceso de desarrollo del ACG. Además, ésto facilita la comprobación y validación de los módulos al compartir raíces comunes.

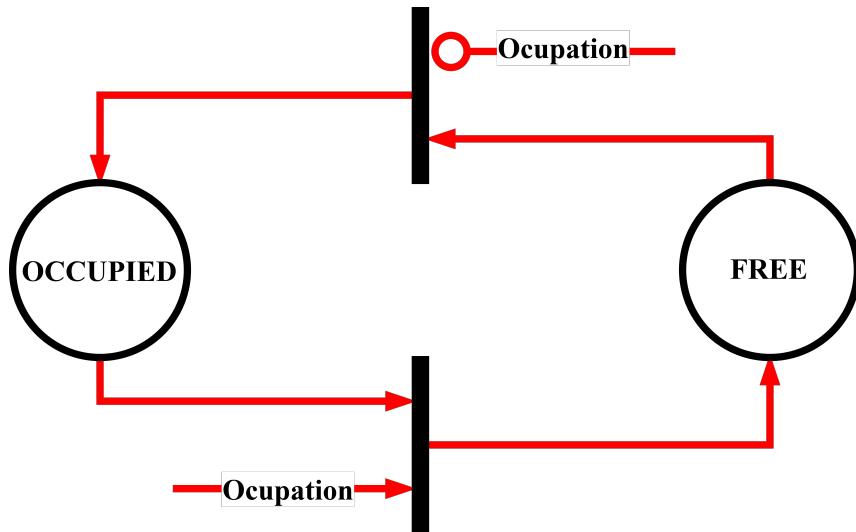
3.2.12. Módulo genérico de los netElements

El módulo *NetElements* (ver Figura 3.14) es el encargado de implementar el reporte del estado de ocupación de las vías. A diferencia de otros módulos centrados en el funcionamiento de mecanismos ferroviarios, que leen el estado del elemento que modelan y envían los comandos necesarios para operarlos; el módulo de *NetElements* solo se ocupa de reportar estados de sólo lectura, la presencia o no de una formación ferroviaria en una vía determinada, sin poder modificar sus valores de ninguna manera.

El ACG utiliza la información otorgada por el RNA para determinar las entradas y salidas cada módulo *NetElements*, e implementa un módulo por cada *NetElement* definido en la red. Cada uno de estos módulos tendrá su propio *NetElement*, leerá su estado (*occupation*) y recibirá consultas de las rutas que los solicitan (*routeCommands*). A su vez, deberán informar a las rutas que los utilizan cuál es su estado (ocupado o libre, en el puerto *state*) y si se encuentran enclavados (liberados, reservados o enclavados; en el puerto *interlock*). El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.35.

Figura 3.35: FSMD del módulo genérico de *NetElements*.

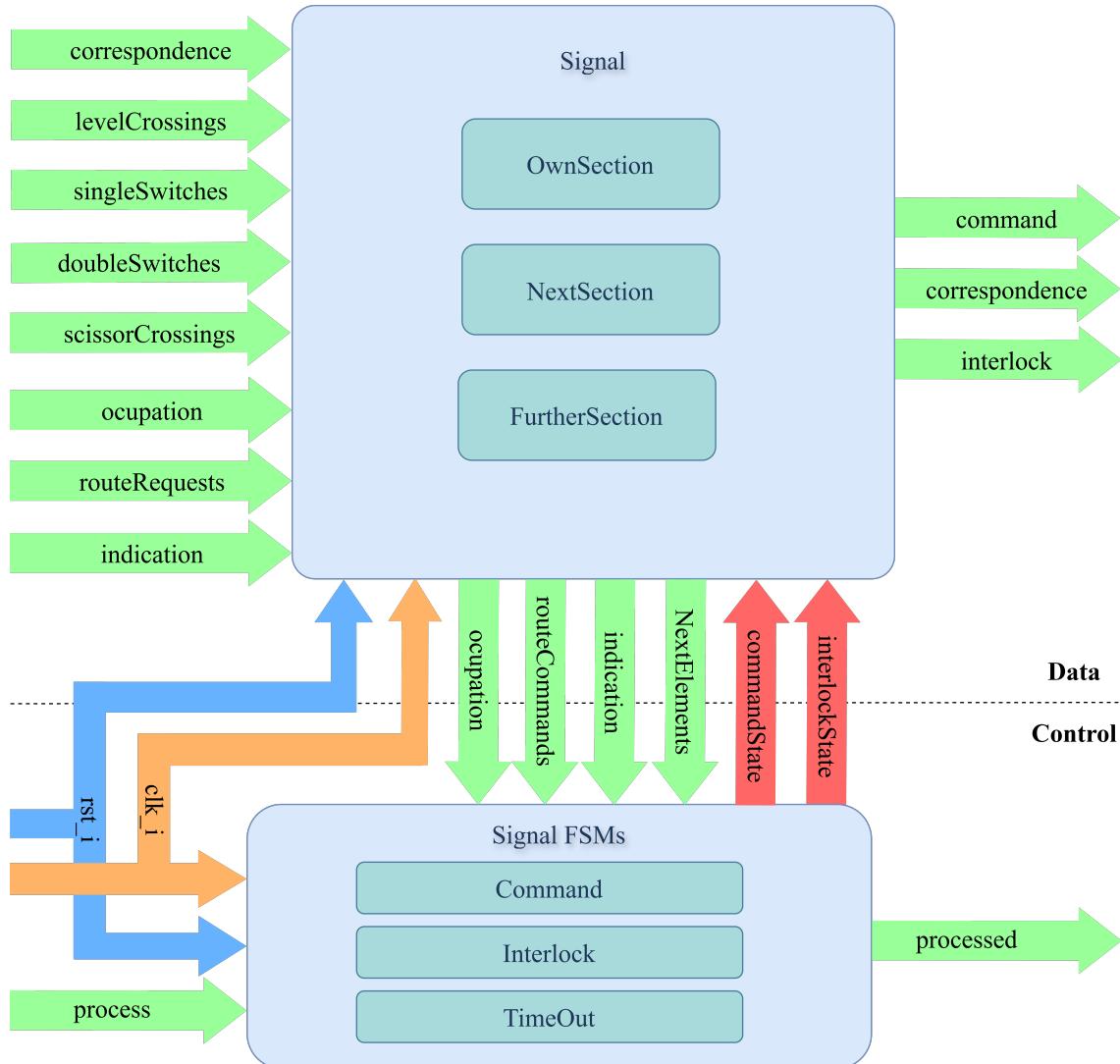
Cualquier ruta que pase por sobre el *NetElement* en cuestión puede consultar su estado utilizando el puerto *routeCommands*. Como ya fue explicado en la Sección 2.1.6, un circuito de vía asociado a un *netElement* puede estar ocupado por una formación o puede estar libre. Todos los módulos *NetElements* reportarán su estado, independientemente si se encuentran reservados o enclavados. La reserva o enclavamiento es meramente para ser utilizado por las rutas para no considerar como propio un *NetElement* que se encuentra libre pero ya ha sido reservado por otra ruta antagónica. El comportamiento del reporte del estado de ocupación de las vías se define en la red de Petri de la Figura 3.36.

Figura 3.36: Red de Petri del modelo dinámico de *NetElements*.

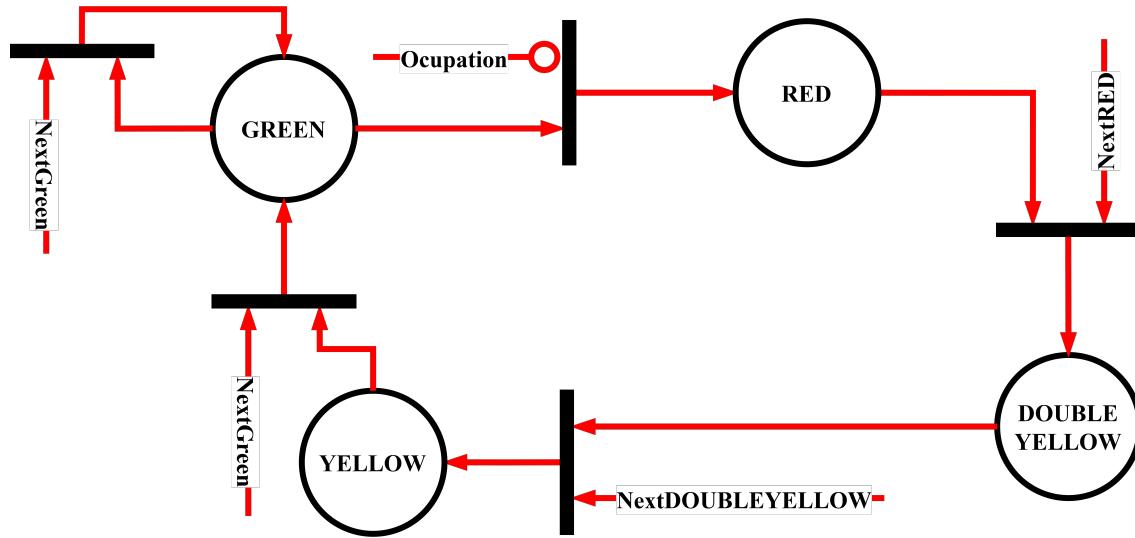
La simpleza de la implementación del módulo *NetElements* facilita una rápida respuesta por parte del sistema de enclavamientos, al otorgar la información requerida solamente a las rutas que la solicitan, lo que reduce enormemente la cantidad de puertos y conexiones a implementar. El ACG crea, además, todas las conexiones necesarias entre cada módulo de elementos ferroviarios y los módulos *NetElements* que los contienen, además de los vecinos más próximos, para disminuir las chances de que ocurran situaciones peligrosas.

3.2.13. Módulo genérico de las señales ferroviarias

El módulo *Signals* (ver Figura 3.14) es el encargado de implementar el funcionamiento de las señales ferroviarias. El ACG determina cuantos caminos posibles existen utilizando como punto de partida la señal ferroviaria a implementar. Cada uno de los caminos posibles influirá en el comportamiento de la señal ferroviaria. Para determinar cual es el único camino posible, el ACG implementa los puertos y conexiones a cada uno de los elementos ferroviarios involucrados con esa señal y hasta dos señales futuras, para todos sus caminos. En base a los estados reportados por estos elementos ferroviarios, el módulo *Signals* determinará cuál es el camino activo y su señal ferroviaria quedará definida totalmente. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.37.

Figura 3.37: FSMD del módulo genérico de *Signals*.

Como fue explicado en la Sección 2.1.10, el aspecto de una señal ferroviaria será rojo si el conjunto de *netElements* al que habilita circular la formación ferroviaria se encuentra ocupado. Si se encontrasen desocupados, su aspecto dependerá del aspecto de la señal siguiente. Cuál es la señal siguiente dependerá del estado actual de toda la infraestructura entre la señal analizada y dos señales consecutivas. El comportamiento del aspecto de una señal ferroviaria se define en la red de Petri de la Figura 3.38.

Figura 3.38: Red de Petri del modelo dinámico de *Signals*.

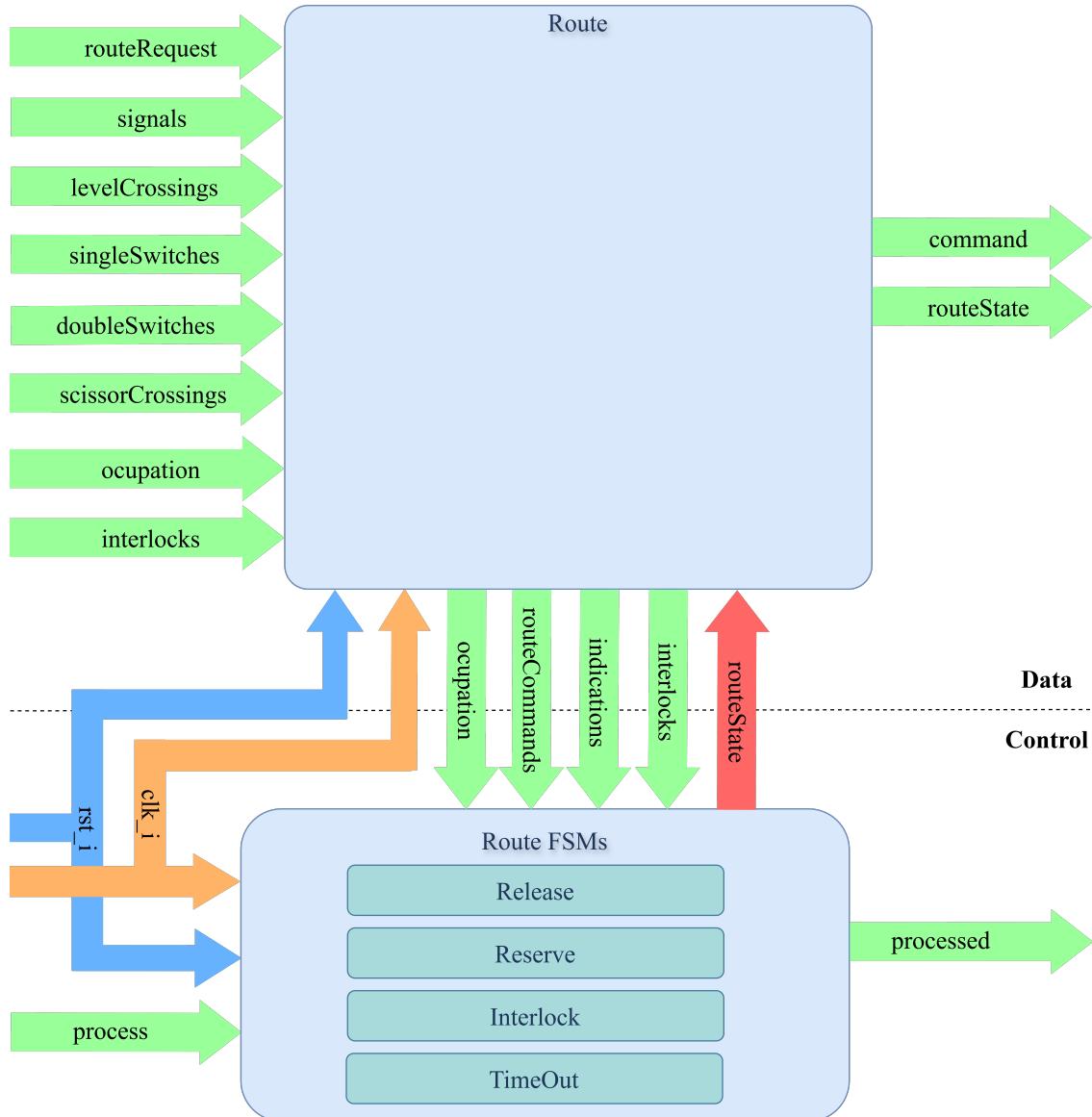
La red de Petri ilustrada en la Figura 3.38 es una simplificación de la implementación real, en la que solamente se ilustra un único cambio de estado entre un aspecto y otro. De haber ilustrado todas las transiciones entre, por ejemplo, el aspecto rojo y amarillo, el aspecto rojo y verde, o el aspecto rojo consigo mismo, entonces la cantidad de transiciones sería tres veces mayor.

Teniendo en cuenta el modelo simplificado, una señal ferroviaria depende de sí misma solo si la sección a la que protege se encuentra ocupada, en cuyo caso su aspecto será rojo. En caso contrario, si la sección se encuentra libre, dependerá del aspecto de la señal siguiente: si la señal siguiente es roja, el aspecto de la señal analizada será doble amarillo; si la señal siguiente es doble amarilla, el aspecto de la señal analizada será amarillo; si la señal siguiente es amarilla, el aspecto de la señal analizada será verde y, finalmente, si la señal siguiente es verde, el aspecto de la señal analizada también lo será.

Las señales ferroviarias no pueden cambiar su aspecto una vez que se encuentran enclavadas, salvo para adoptar un aspecto más restrictivo, para garantizar un mayor nivel de seguridad.

3.2.14. Módulo genérico de las rutas ferroviarias

El módulo *Routes* (ver Figura 3.14) es el encargado de implementar el funcionamiento de las rutas ferroviarias. El ACG determina todos los elementos ferroviarios que abarca la ruta e implementa los puertos y conexiones a cada uno de estos elementos. De esta manera, cada módulo *Routes* tiene la información necesaria del estado actual de cada elemento y si se encuentran o no enclavados. Además, el ACG implementa las salidas necesarias para consultar y controlar cada elemento ferroviario en cuestión, junto con la dinámica interna para determinar si la ruta puede ser habilitada y bajo qué condiciones. El diagrama de bloques de la máquina de estados finitos con camino de datos diseñado para lograr este objetivo se muestra en la Figura 3.39.

Figura 3.39: FSMD del módulo genérico de *Routes*

Inicialmente, el módulo *Routes* se encuentra inactivo, esperando que la ruta sea solicitada por el operador. De recibir la orden esperada, el módulo comprobará que todos los *netElements* se encuentran liberados (ni reservados ni enclavados por otras rutas) y libres (no ocupados por ninguna formación). De ser afirmativa la respuesta, se enviará la señal de reserva y de confirmarse la reserva, se envía la señal de enclavamiento a los *netElements*. Estos pasos deberán hacerse dentro de un intervalo de tiempo determinado o el pedido de ruta será cancelado.

A continuación, una vez se confirma que toda la sección está enclavada, se repite el procedimiento de verificación de estado, reserva y enclavamiento con cada elemento ferroviario necesario para asegurar la ruta. Si alguno de estos elementos no se encuentra disponible, la ruta es cancelada por timeout y todas las secciones y componentes previamente enclavados son liberados. Si todas las condiciones de ruta son satisfechas, la ruta es habilitada y se toma el control de la señal inicial.

El comportamiento de las rutas ferroviarias se define en la red de Petri de la Figura 3.40.

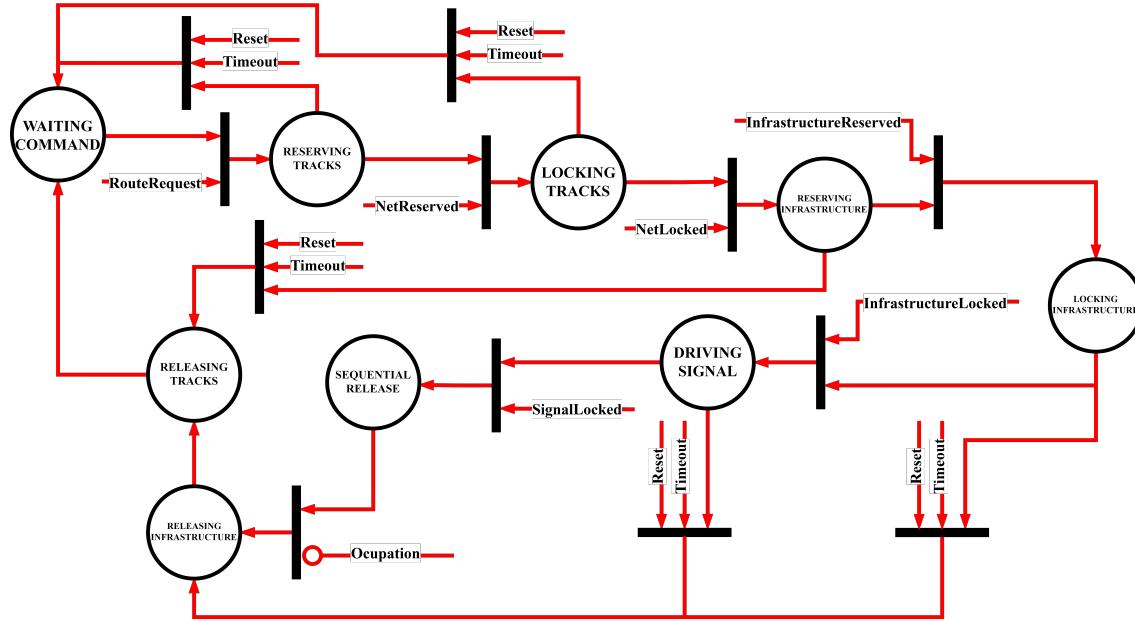


Figura 3.40: Red de Petri del modelo dinámico de *Routes*.

A medida que la formación va circulando se activa la liberación secuencial de los elementos ferroviarios dejados atrás por la formación, lo cual permite una mayor flexibilidad a la hora de asignar y liberar rutas, como se explicó en la Sección 3.1.6. Al finalizar la liberación secuencial, si la formación ya no ocupa ninguna sección de la ruta, entonces se procede a liberar los elementos restantes y la ruta vuelve a su estado de espera inicial.

Las rutas no comprueban el estado de rutas antagónicas ni rutas con las que comparten recursos, solamente sus propias condiciones de habilitación. Esto es así ya que se delegó en cada elemento la facultad de ser reservados y enclavados por una única ruta, lo cual impide la existencia de rutas conflictivas habilitadas simultáneamente. De otra manera, las rutas usarían demasiados recursos mientras que los elementos ferroviarios mucho menos. Una distribución de las responsabilidades otorga más flexibilidad y escalabilidad al sistema, con módulos de similares tamaños y complejidad.

3.3. Redundancia

Para lograr una mayor fiabilidad en los resultados e incrementar la seguridad del sistema, el módulo *Network* (ver Sección 3.2.7) fue instanciado tres veces e interconectado con un sistema de votación 2oo3, tal como se explicó en la Sección 1.4.2.

3.4. Plataforma utilizada

El código generado por el ACG es independiente de la plataforma que se utilice. Por cuestiones de disponibilidad y experiencia previa con la plataforma durante la Maestría de Sistemas Embebidos, se seleccionó la FPGA ARTY Z7-20 de Xilinx (Figura 3.41). La misma posee 53200

Look-up-tables, 106400 Flip-Flops, 32 Buffers y 125 bloques de entrada/salida. El precio promedio de esta plataforma ronda los 300 U\$D, un precio razonable para una plataforma de nivel intermedio.

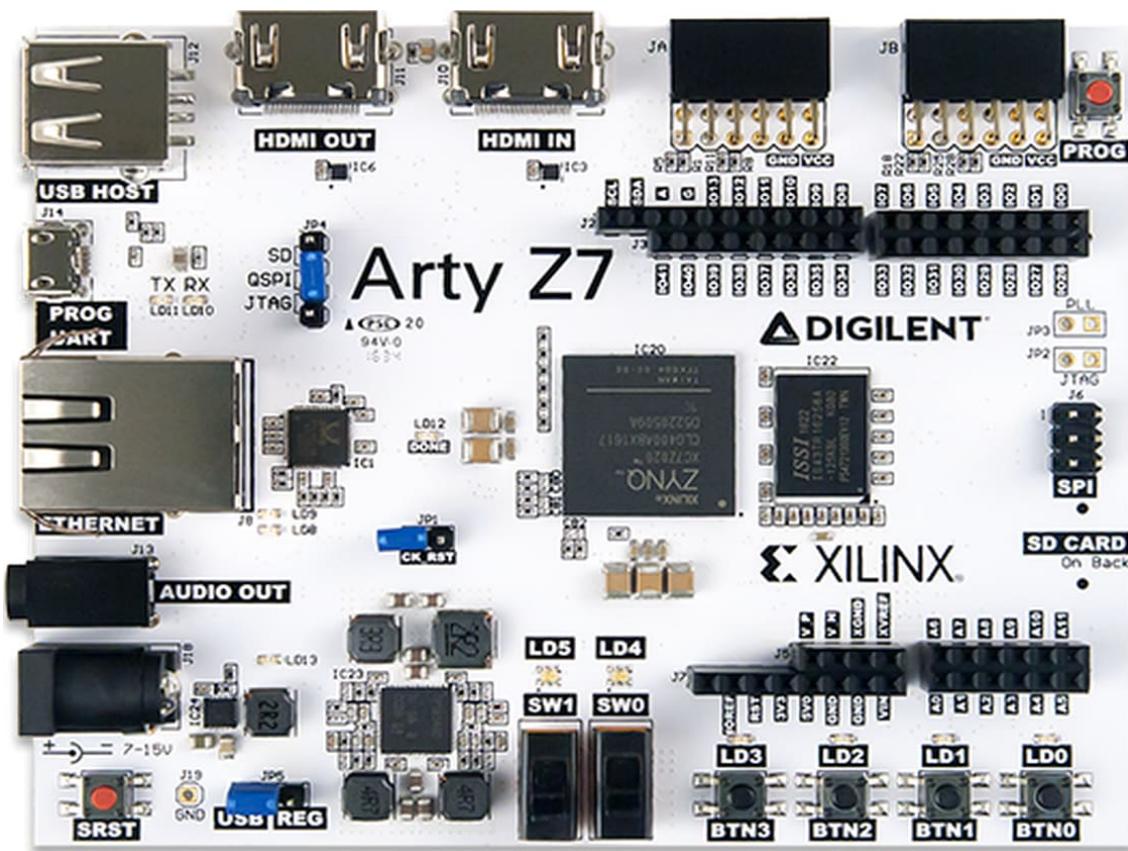


Figura 3.41: Plataforma FPGA Xilinx Arty Z7-20.

Aunque se seleccionó una FPGA de la familia Xilinx [30, 35, 46, 47, 52, 75, 91, 248, 249] y se utilizó el entorno de desarrollo integrado Vivado [250, 251], el código VHDL generado por el ACG es independiente de la plataforma que se utilice. De emplearse otra familia de FPGAs, solo será necesaria una nueva asignación manual de pines de entrada y salida. De emplearse FPGAs por fuera de las familias ofrecidas por Xilinx, como por ejemplo Intel, puede utilizarse el entorno de desarrollo Inter Quartus [252].

Tanto el entorno de desarrollo Xilinx Vivado como el Intel Quartus incorporan herramientas para validación de la sintaxis generada por el ACG. Se utilizó la primera de ellas para generar el diagrama de bloques del sistema y analizar cada uno de los módulos y sus elementos internos. Este proceso incluye una validación de la sintaxis de todos los archivos generados por el ACG. Un análisis similar se realiza en las etapas de síntesis e implementación, detallado en profundidad en el Capítulo 4.

La elección de una FPGA por sobre un microprocesador se debe a las ventajas expuestas en la Sección 1.3.1 respecto a la concurrencia del sistema, mayor nivel de seguridad y facilidad para redundar el sistema. Además, los sistemas implementados en FPGA son considerados puramente hardware y no software, por lo que solamente deben cumplir los requerimientos de la norma EN 50129, especialmente el anexo F referido a FPGAs, y no la norma EN 50128 que se encarga del software. Los alcances de estas normas ya fueron explicados en la Sección 1.4.1.

Capítulo 4

Resultados obtenidos

Se utilizó el RNA y el ACG en nueve diferentes ejemplos. Estos ejemplos abarcan sistemas simples y complejos, desde ejemplos provistos por railML.org hasta sistemas reales, pasando por locaciones generadas artificialmente por mi para probar cada una de las funcionalidades. En las siguientes secciones se analizará a detalle el primer ejemplo. Los ocho ejemplos restantes se encuentran en los apéndices correspondientes.

El hecho de que el ejemplo seleccionado cuente con un señalamiento ya establecido permitirá obtener conclusiones respecto a la eficacia del RNA, su impacto en la logística, seguridad y cobertura de la red. Es esperable que, habiendo probado que el señalamiento generado por el RNA es tan o mas efectivo que un señalamiento tradicional, se obtengan resultados comparables a la hora de generar señalamiento para locaciones que cuenten con un señalamiento limitado o nulo.

El análisis incluye la presentación de la topología original junto con su señalamiento y tabla de enclavamientos, de existir. Adicionalmente, se ilustrará la generación paso a paso del señalamiento, su simplificación y tabla de enclavamientos final, junto con la comparación con el enclavamiento original, si existiese. Finalmente, se incluyen las redes de grafos generadas por el RNA y un resumen de los recursos utilizados por el ACG para implementar el sistema.

4.1. Ejemplo 1

4.1.1. Topología ferroviaria original

El primer ejemplo, ilustrado en la Figura 4.1, es una topología diseñada en base a dos líneas principales y tres niveles de ramificaciones. La primera ramificación, utilizando el cambio de vías Sw06, es una ramificación simple. La segunda ramificación, utilizando el cambio de vías Sw04, es una ramificación compleja al incluir el cambio de vías Sw07 a continuación. Ademas, se incluyeron los cambios de vías Sw12 y Sw13 para permitir el intercambio de formaciones entre ambas vías principales. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología de múltiples ramificaciones anidadas.

Para incrementar la dificultad del análisis y obtener resultados mas completos, se incluyeron finales de vías relativos y absolutos, junto con plataformas (Platform11, Platform13) y cruces de vías (LevelCrossing06, LevelCrossing 08). La infraestructura se distribuyó de forma tal que no en todos los casos existiese espacio suficiente entre los elementos ferroviarios. Por ejemplo, entre la plataforma Platform11 y el cruce de vía LevelCrossing6 existe el espacio suficiente, en cambio entre la plataforma Platform13 y el cruce de vía LevelCrossing8 no existe tanto espacio entre ellos.

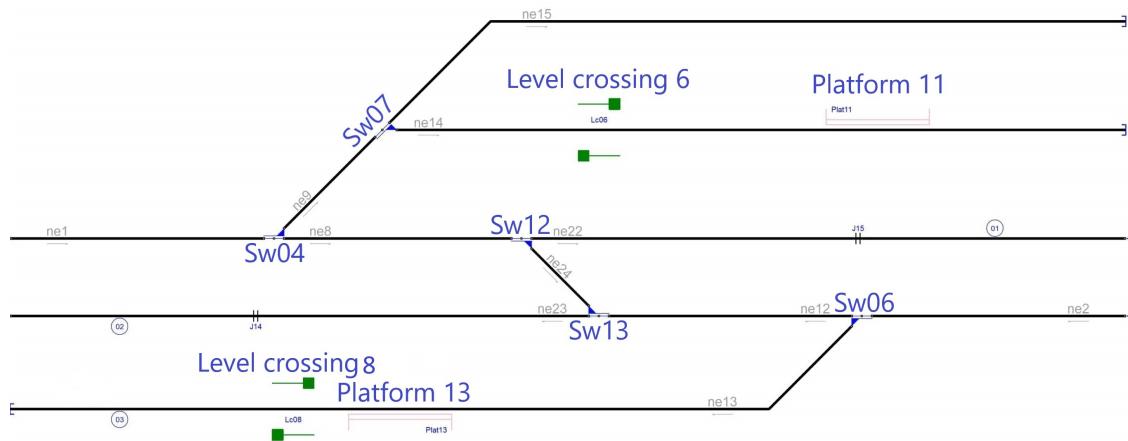


Figura 4.1: Topología ferroviaria del ejemplo 1 sin señalamiento.

4.1.2. Señalamiento original

El señalamiento original, ilustrado en la Figura 4.2, incluye señales de parada próximas a los finales de vías absolutos (S18, S19, S20), señales de partida en las plataformas (S04, S06, S08, S09), señales de protección antes de cada paso a nivel (S04, S05), señales de maniobras antes de converger en una vía principal (S03, S08) y señales múltiples para cambios de vías divergentes (S01, S17, S10, S07), entre varias otras señales.

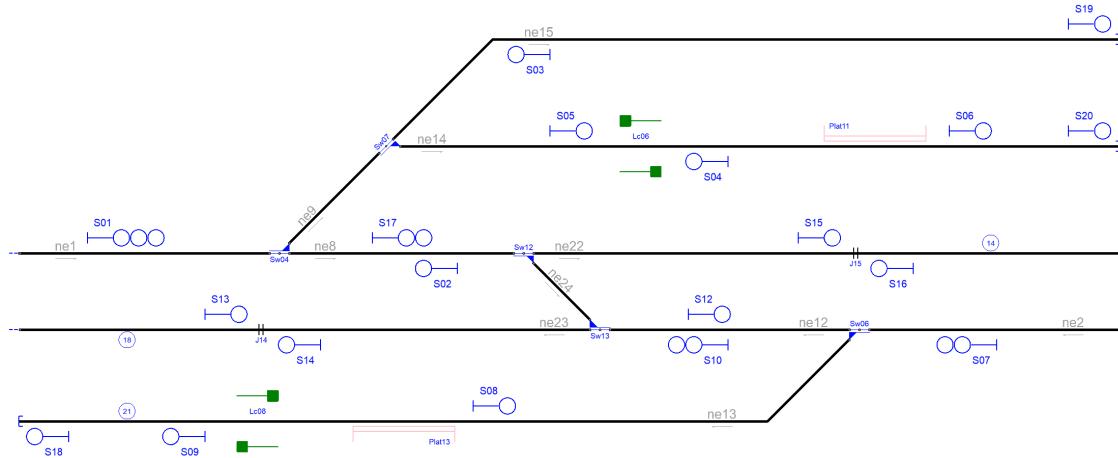


Figura 4.2: Señalamiento original del ejemplo 1.

Estas señales permiten definir hasta un máximo de 14 rutas, todas ellas detalladas en la Tabla 4.1. En una primera inspección, se puede comprobar que todos los elementos ferroviarios son alcanzados por al menos una de las rutas, en al menos una dirección. Además, todos los cambios de vías son utilizados, de forma simple o compuesta.

Tabla 4.1: Tabla de enclavamiento original del ejemplo 1.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	S ₀₅	S ₀₆	-	Plat ₁₁	Lc ₀₆	ne ₁₄
R ₀₂	S ₀₆	S ₂₀	-	-	-	ne ₁₄
R ₀₃	S ₀₉	S ₁₈	-	-	-	ne ₁₃
R ₀₄	S ₁₃	S ₁₂	Sw ₁₃ ^N	-	-	ne ₂₃ -ne ₁₂
R ₀₅	S ₁₆	S ₀₂	Sw ₁₂ ^N	-	-	ne ₂₂ -ne ₀₈
R ₀₆	S ₀₇	S ₁₀	Sw ₀₆ ^N	-	-	ne ₀₂ -ne ₁₂
R ₀₇	S ₀₇	S ₀₉	Sw ₀₆ ^R	Plat ₁₃	Lc ₀₈	ne ₀₂ -ne ₁₃
R ₀₈	S ₁₀	S ₁₄	Sw ₁₃ ^N	-	-	ne ₁₂ -ne ₂₃
R ₀₉	S ₁₀	S ₀₂	Sw ₁₂ ^R +Sw ₁₃ ^R	-	-	ne ₁₂ -ne ₂₄ -ne ₀₈
R ₁₀	S ₀₁	S ₁₇	Sw ₀₄ ^N	-	-	ne ₀₁ -ne ₀₈
R ₁₁	S ₀₁	S ₁₉	Sw ₀₄ ^R +Sw ₀₇ ^N	-	-	ne ₀₁ -ne ₁₅
R ₁₂	S ₀₁	S ₀₅	Sw ₀₄ ^R +Sw ₀₇ ^R	-	-	ne ₀₁ -ne ₁₄
R ₁₃	S ₁₇	S ₁₅	Sw ₁₂ ^N	-	-	ne ₀₈ -ne ₂₂
R ₁₄	S ₁₇	S ₁₂	Sw ₁₂ ^R +Sw ₁₃ ^R	-	-	ne ₀₈ -ne ₂₄ -ne ₁₂

Algunas rutas abarcan mas de un *netElement*, como por ejemplo la ruta R14 que comienza en la señal S17 y finaliza en la señal S12, atravesando los *netElements* ne8, ne24 y ne12, utilizando los cambios de vías Sw12 y Sw13, ambos en posición reversa.

4.1.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código 4.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código 4.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analysing railML object
Analysing graph
ne1 [810, 150] [1320, 150] >>
ne2 [2970, 0] [2460, 0] <<
ne8 [1320, 150] [1800, 150] >>
ne9 [1320, 150] [1530, 360] >>
ne12 [2460, 0] [1950, 0] <<
ne13 [2460, 0] [810, -180] <<
ne14 [1530, 360] [2970, 360] >>
ne15 [1530, 360] [2970, 570] >>
ne22 [1800, 150] [2970, 150] >>
ne23 [1950, 0] [810, 0] <<
ne24 [1800, 150] [1950, 0] >>
```

The network is connected

Por ejemplo, el *netElement* ne1 inicia en la coordenada (810;150) y finaliza en la coordenada (1320;150). El símbolo \gg indica que ne1 se encuentra definido de izquierda a derecha, ya que la componente x de la coordenada final es mayor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura 4.2. Por ejemplo, ne1, ne8 y ne9 comparten la coordenada (1320;150), que coincide con la coordenada del cambio de vías Sw04.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código 4.2.

Código 4.2: Detección de puntos críticos por parte del RNA

```
Analysing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne1 has a Middle point @ [1065.0, 150]
ne2 has a Middle point @ [2715.0, 0]
ne8 has a Middle point @ [1560.0, 150]
ne12 has a Middle point @ [2205.0, 0]
ne13 has a Platform[plf75] @ [1564, 180]
ne13 has a LevelCrossing[lcr74] @ [1362, 180]
ne13 has a Curve(2 lines) @ [[2280, -180]]
ne14 has a Platform[plf68] @ [2490, -360]
ne14 has a LevelCrossing[lcr69] @ [1945, -360]
ne15 has a Curve(2 lines) @ [[1740, 570]]
ne22 has a RailJoint[J15] @ [2452, 150]
ne23 has a RailJoint[J14] @ [1284, 0]
```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código F.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código 4.3: Infrastructure.RNA

```
Nodes: 11|Switches: 5|Signals: 0|Detectors: 2|Ends: 7|Barriers: 2
Node ne1:
    Track = track1
    Neighbours = 2 -> ['ne8', 'ne9']
    Switches -> Sw04
        ContinueCourse -> right -> ne8
        BranchCourse -> left -> ne9
Node ne2:
    Track = track7
    Neighbours = 2 -> ['ne12', 'ne13']
    Switches -> Sw06
        ContinueCourse -> right -> ne12
        BranchCourse -> left -> ne13
Node ne8:
```

```

Track = track2
Neighbours = 4 -> ['ne1', 'ne9', 'ne22', 'ne24']
Switches -> Sw12
    ContinueCourse -> left -> ne22
    BranchCourse -> right -> ne24

Node ne9:
    Track = track3
    Neighbours = 4 -> ['ne1', 'ne8', 'ne14', 'ne15']
    Switches -> Sw07
        ContinueCourse -> left -> ne15
        BranchCourse -> right -> ne14

Node ne12:
    Track = track8
    Neighbours = 4 -> ['ne2', 'ne13', 'ne23', 'ne24']
    Switches -> Sw13
        ContinueCourse -> left -> ne23
        BranchCourse -> right -> ne24

Node ne13:
    Track = track9
    Type = BufferStop -> ['bus56']
    Neighbours = 2 -> ['ne2', 'ne12']
    Level crossing -> lcr74
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [1317, 180] | Coordinate: 0.7059

Node ne14:
    Track = track4
    Type = BufferStop -> ['bus10']
    Neighbours = 2 -> ['ne9', 'ne15']
    Level crossing -> lcr69
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [1990, -360] | Coordinate: 0.3197

Node ne15:
    Track = track10
    Type = BufferStop -> ['bus59']
    Neighbours = 2 -> ['ne9', 'ne14']

Node ne22:
    Track = track6
    TrainDetectionElements -> tde78
    Type -> insulatedRailJoint
    Neighbours = 2 -> ['ne8', 'ne24']

Node ne23:
    Track = track5
    TrainDetectionElements -> tde77
    Type -> insulatedRailJoint
    Neighbours = 2 -> ['ne12', 'ne24']

Node ne24:
    Track = track11
    Neighbours = 4 -> ['ne8', 'ne12', 'ne22', 'ne23']

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código 4.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código 4.4: SafePoints.RNA

```

ne1:
    Next: [[1065.0, 150]]
    Prev: [[1065.0, 150]]
ne2:
    Next: [[2715.0, 0]]
    Prev: [[2715.0, 0]]
ne8:
    Next: [[1560.0, 150]]
    Prev: [[1560.0, 150]]
ne12:
    Next: [[2205.0, 0]]
    Prev: [[2205.0, 0]]
ne13:
    Next: [[1162, 180], [2180.0, -180]]
    Prev: [[1764, 180]]
ne14:
    Next: [[2290, -360], [1745, -360]]
    Prev: [[2690, -360], [2145, -360]]
ne15:
    Prev: [[1840.0, 570]]
ne22:
    Next: [[2352.0, 150]]
    Prev: [[2552.0, 150]]
ne23:
    Next: [[1184.0, 0]]
    Prev: [[1384.0, 0]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura 4.3.

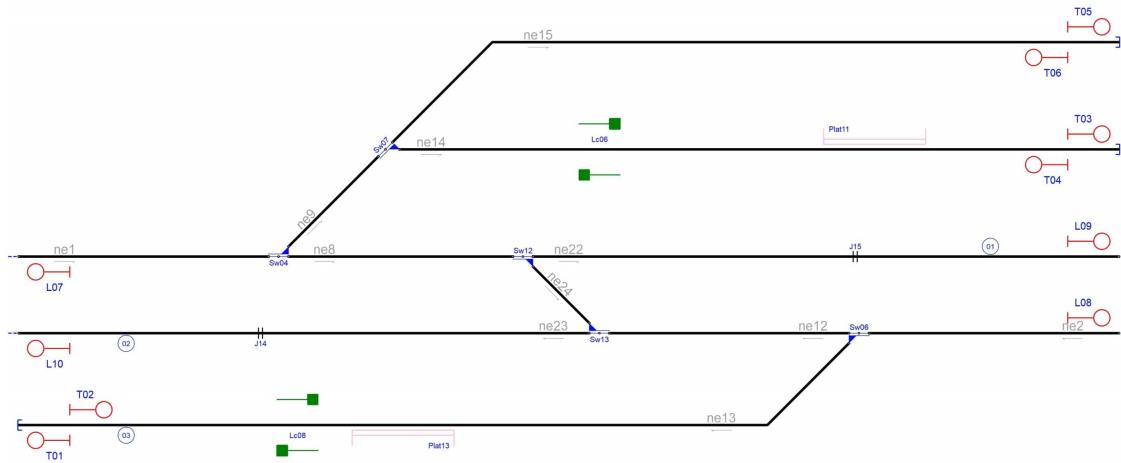


Figura 4.3: Señalamiento generado por el RNA para proteger el fin de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01, T03, T05 y las señales de partida son T02, T04 y T06. A su vez, los finales de vías relativos poseen las señales de parada L07, L08, L09 y L10, cercanos al límite del externo del *netElement* al que pertenecen.

La Figura 4.4 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Estas señales se obtuvieron al aplicar el Algoritmo 8, tal como fue explicado en la Sección 2.2.2. Las señales generadas son J11, J12, J13 y J14, indicadas en color rojo. De no existir junturas que proteger, el RNA saltará este paso.

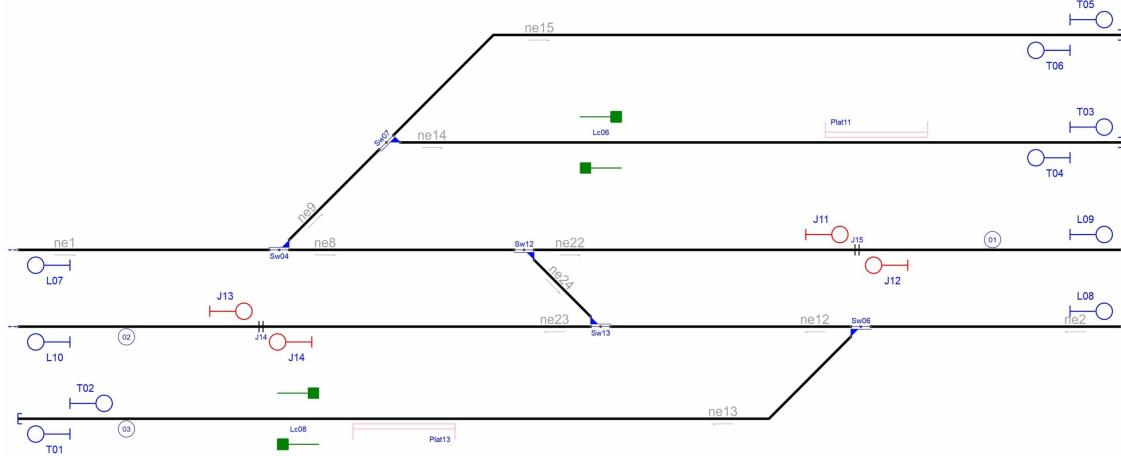


Figura 4.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos, tal como sucede con los elementos levelCrossing8 y Platform13. El señalamiento generado para proteger las plataformas y los cruces de vía, producto de aplicar el Algoritmo 9 y el Algoritmo 10, respectivamente, se ilustra en rojo en la Figura 4.5. Las señales generadas para proteger las plataformas son las señales de partida P18, P19 y P20, mientras que las señales que protegen los cruces de vía son las señales X15, X16 y X17.

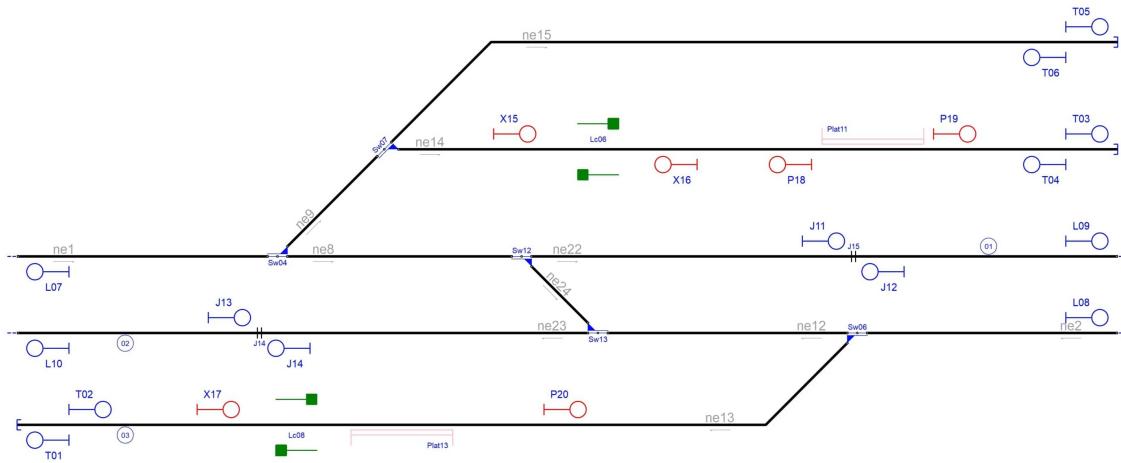


Figura 4.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA genera las señales S22, C21, H23 y H24 para proteger el cambio de vías Sw04; las señales S27, C25, B26 y H28 para proteger el cambio de vías Sw06; las señales C29 y B30 para proteger el cambio de vías Sw07; las señales S32, C31 y H33 para proteger el cambio de vías Sw12 y las señales S35, C34 y H36 para proteger el cambio de vías Sw13. Estas señales se encuentran resaltadas en rojo en la Figura 4.6.

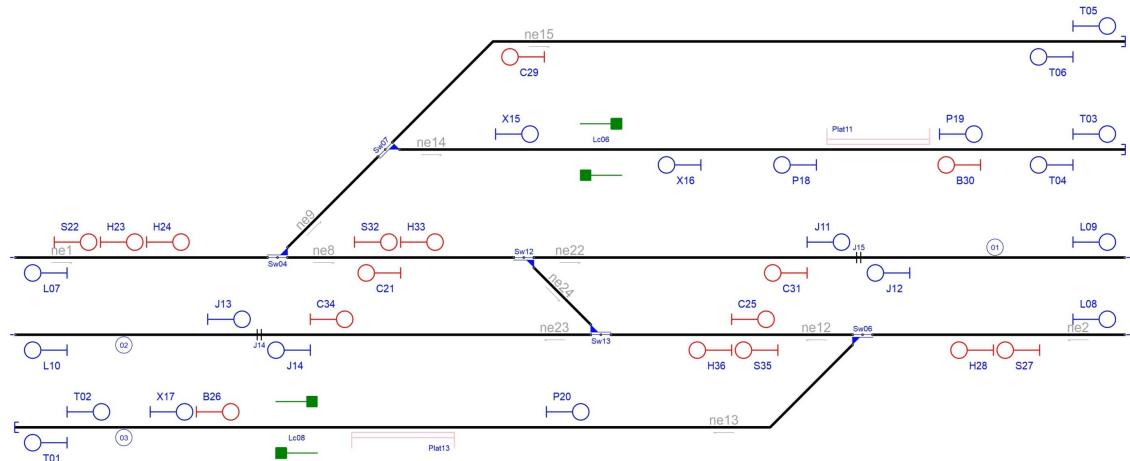


Figura 4.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías Sw12 y Sw13, desplazando las señales hasta convertirlas en las señales H33 y H36 respectivamente. Análogamente, las señales C y S del *netElement* se convirtieron en las señales H23 y H24 respectivamente.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: X17, P18, P19, B26, B30, C31 y C34. Las señales X17 y B26 fueron eliminadas por su cercanía con la señal T02,

con la cual comparten dirección y sentido. Lo mismo ocurre entre el par de señales P18/B30 y la señal T04; entre las señales P19 y T03; entre las señales C31 y J12; y entre las señales C34 y J13. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, y así generar el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código 4.5.

Código 4.5: Reducción de señalamiento por prioridad de señales

```
Reducing redundant signals
removing sig17 for sig02
removing sig26 for sig02
removing sig19 for sig03
removing sig30 for sig04
removing sig31 for sig12
removing sig34 for sig13
removing sig18 for sig30
```

El resultado de la simplificación del señalamiento se ilustra en la Figura 4.7.

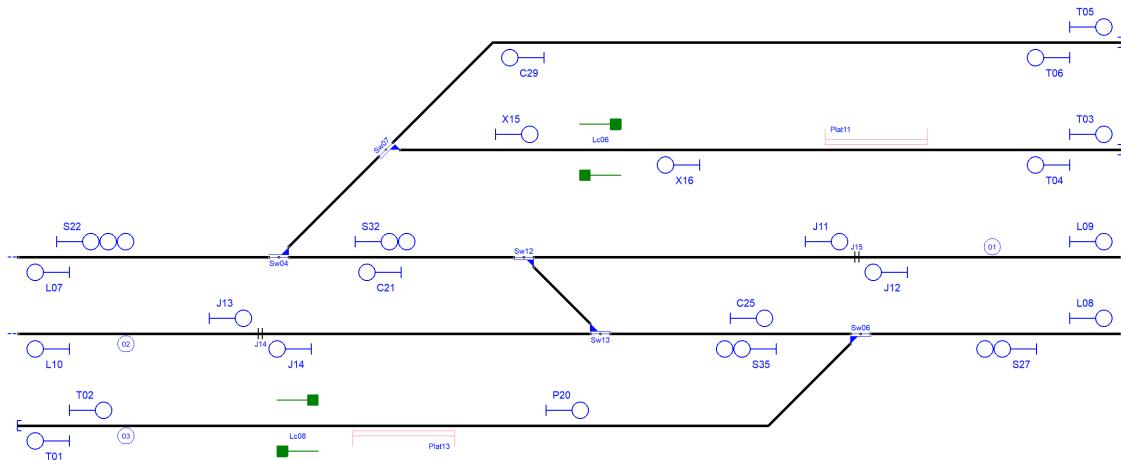


Figura 4.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código 4.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código 4.6: Signalling.RNA

```
T01 [T01] <<:
  From: ne13 | To: bus56_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [910, 180] | Coordinate: 0.2055
T02 [T02] >>:
  From: ne13 | To: ne13_right
  Type: Stop | Direction: reverse | AtTrack: right
```

```
Position: [910, 180] | Coordinate: 0.2055
T03 [T03] >>:
    From: ne14 | To: bus10_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [2870, -360] | Coordinate: 0.9305
T04 [T04] <<:
    From: ne14 | To: ne14_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [2870, -360] | Coordinate: 0.9305
T05 [T05] >>:
    From: ne15 | To: bus59_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [2870, -570] | Coordinate: 0.9345
T06 [T06] <<:
    From: ne15 | To: ne15_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [2870, -570] | Coordinate: 0.9345
L07 [L07] <<:
    From: ne1 | To: oe40_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [910, -150] | Coordinate: 0.1960
L08 [L08] >>:
    From: ne2 | To: oe42_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [2870, 0] | Coordinate: 0.8039
L09 [L09] >>:
    From: ne22 | To: oe41_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [2870, -150] | Coordinate: 0.9145
J11 [J11] >>:
    From: ne22 | To: ne22_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [2352.0, -150] | Coordinate: 0.4717
J12 [J12] <<:
    From: ne22 | To: ne22_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [2552.0, -150] | Coordinate: 0.6427
J13 [J13] >>:
    From: ne23 | To: ne23_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1184.0, 0] | Coordinate: 0.3280
J14 [J14] <<:
    From: ne23 | To: ne23_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1384.0, 0] | Coordinate: 0.5035
X15 [X15] >>:
    From: ne14 | To: ne14_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1745, 360] | Coordinate: 0.5218
X16 [X16] <<:
    From: ne14 | To: ne14_left
```

```

Type: Circulation | Direction: reverse | AtTrack: right
Position: [2145, 360] | Coordinate: 0.6575
P20 [P20] >>:
    From: ne13 | To: ne13_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1844, -180] | Coordinate: 0.7824
C21 [C21] <<:
    From: ne8 | To: ne8_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1560.0, -150] | Coordinate: 0.5
S22 [S22] >>:
    From: ne1 | To: ne1_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1065.0, -150] | Coordinate: 0.5
C25 [C25] >>:
    From: ne12 | To: ne12_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [2205.0, 0] | Coordinate: 0.5
S27 [S27] <<:
    From: ne2 | To: ne2_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [2715.0, 0] | Coordinate: 0.5
C29 [C29] <<:
    From: ne15 | To: ne15_left
    Type: Manouver | Direction: reverse | AtTrack: right
    Position: [1840.0, -570] | Coordinate: 0.2599
S32 [S32] >>:
    From: ne8 | To: ne8_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1560.0, -150] | Coordinate: 0.5
S35 [S35] <<:
    From: ne12 | To: ne12_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [2205.0, 0] | Coordinate: 0.5

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código 4.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código 4.7: Routes.RNA

```

route_1 [T02 >> P20]:
    Path: ['ne13']
    Platforms: ['Plat13']
    LevelCrossings: ['Lc08']
route_2 [T04 << X16]:
    Path: ['ne14']
    Platforms: ['Plat11']
route_3 [T06 << C29]:

```

```
    Path: ['ne15']
route_4 [J11 >> L09]:
    Path: ['ne22']
route_5 [J12 << C21]:
    Path: ['ne22', 'ne8']
    Switches: ['Sw12_N']
route_6 [J13 >> C25]:
    Path: ['ne23', 'ne12']
    Switches: ['Sw13_N']
route_7 [X15 >> T03]:
    Path: ['ne14']
    Platforms: ['Plat11']
    LevelCrossings: ['Lc06']
route_8 [X16 << L07]:
    Path: ['ne14', 'ne9', 'ne1']
    Switches: ['Sw04_R', 'Sw07_R']
    LevelCrossings: ['Lc06']
route_9 [P20 >> L08]:
    Path: ['ne13', 'ne2']
    Switches: ['Sw06_R']
route_10 [C21 << L07]:
    Path: ['ne8', 'ne1']
    Switches: ['Sw04_N']
route_11 [S22 >> S32]:
    Path: ['ne1', 'ne8']
    Switches: ['Sw04_N']
route_12 [S22 >> X15]:
    Path: ['ne1', 'ne9', 'ne14']
    Switches: ['Sw04_R', 'Sw07_R']
route_13 [S22 >> T05]:
    Path: ['ne1', 'ne9', 'ne15']
    Switches: ['Sw04_R', 'Sw07_N']
route_14 [C25 >> L08]:
    Path: ['ne12', 'ne2']
    Switches: ['Sw06_N']
route_15 [S27 << S35]:
    Path: ['ne2', 'ne12']
    Switches: ['Sw06_N']
route_16 [S27 << T01]:
    Path: ['ne2', 'ne13']
    Switches: ['Sw06_R']
    Platforms: ['Plat13']
    LevelCrossings: ['Lc08']
route_17 [C29 << L07]:
    Path: ['ne15', 'ne9', 'ne1']
    Switches: ['Sw04_R', 'Sw07_N']
route_18 [S32 >> J11]:
    Path: ['ne8', 'ne22']
    Switches: ['Sw12_N']
route_19 [S32 >> C25]:
    Path: ['ne8', 'ne24', 'ne12']
```

```

    Switches: ['Sw12_R', 'Sw13_R']
route_20 [S35 << J14]:
    Path: ['ne12', 'ne23']
    Switches: ['Sw13_N']
route_21 [S35 << C21]:
    Path: ['ne12', 'ne24', 'ne8']
    Switches: ['Sw12_R', 'Sw13_R']

```

4.1.4. Red de grafos generada por el RNA

La información exportada en el Código F.3 (Infrastructure.RNA) Código 4.4 (SafePoint.RNA), Código 4.6 (Signalling.RNA) y Código 4.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura 4.8.

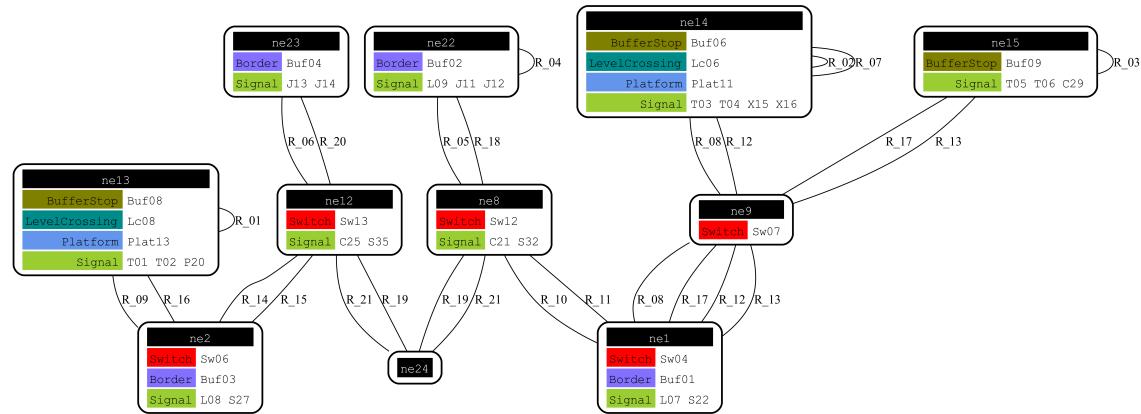


Figura 4.8: Red de grafos generada por el RNA para el ejemplo 1.

Cada nodo del grafo de la Figura 4.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

4.1.5. Señalamiento generado por el RNA

El RNA también exporta la información mostrada en el Código 4.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla 4.2.

Tabla 4.2: Tabla de enclavamiento del ejemplo 1 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	P ₂₀	-	Plat ₁₃	Lc ₀₈	ne ₁₃
R ₀₂	T ₀₄	X ₁₆	-	Plat ₁₁	-	ne ₁₄
R ₀₃	T ₀₆	C ₂₉	-	-	-	ne ₁₅
R ₀₄	J ₁₁	L ₀₉	-	-	-	ne ₂₂
R ₀₅	J ₁₂	C ₂₁	Sw ₁₂ ^N	-	-	ne ₂₂ -ne ₀₈
R ₀₆	J ₁₃	C ₂₅	Sw ₁₃ ^N	-	Lc ₀₈	ne ₂₃ -ne ₁₂
R ₀₇	X ₁₅	T ₀₃	-	Plat ₁₁	Lc ₀₆	ne ₁₄
R ₀₈	X ₁₆	L ₀₇	Sw ₀₄ ^R +Sw ₀₇ ^R	-	-	ne ₁₄ -ne ₀₁
R ₀₉	P ₂₀	L ₀₈	Sw ₀₆ ^R	-	-	ne ₁₃ -ne ₀₂
R ₁₀	C ₂₁	L ₀₇	Sw ₀₄ ^N	-	-	ne ₀₈ -ne ₀₁
R ₁₁	S ₂₂	S ₃₂	Sw ₀₄ ^N	-	-	ne ₀₁ -ne ₀₈
R ₁₂	S ₂₂	X ₁₅	Sw ₀₄ ^R +Sw ₀₇ ^R	-	-	ne ₀₁ -ne ₁₄
R ₁₃	S ₂₂	T ₀₅	Sw ₀₄ ^R +Sw ₀₇ ^N	-	-	ne ₀₁ -ne ₁₅
R ₁₄	C ₂₅	L ₀₈	Sw ₀₆ ^N	-	-	ne ₁₂ -ne ₀₂
R ₁₅	S ₂₇	S ₃₅	Sw ₀₆ ^N	-	-	ne ₀₂ -ne ₁₂
R ₁₆	S ₂₇	T ₀₁	Sw ₀₆ ^R	Plat ₁₃	Lc ₀₈	ne ₀₂ -ne ₁₃
R ₁₇	C ₂₉	L ₀₇	Sw ₀₄ ^R +Sw ₀₇ ^N	-	-	ne ₁₅ -ne ₀₁
R ₁₈	S ₃₂	J ₁₁	Sw ₁₂ ^N	-	-	ne ₀₈ -ne ₂₂
R ₁₉	S ₃₂	C ₂₅	Sw ₁₂ ^R +Sw ₁₃ ^R	-	-	ne ₀₈ -ne ₁₂
R ₂₀	S ₃₅	J ₁₄	Sw ₁₃ ^N	-	-	ne ₁₂ -ne ₂₃
R ₂₁	S ₃₅	C ₂₁	Sw ₁₂ ^R +Sw ₁₃ ^R	-	-	ne ₁₂ -ne ₀₈

En una primera inspección podemos ver que el nuevo señalamiento tiene 21 rutas, versus las 14 rutas del señalamiento original (ver Tabla 4.1). Esto se debe a que todas las vías son consideradas de ambos sentidos por el RNA, lo cuál queda de manifiesto cuando se comprueba que todas las plataformas y cruces de vía son atravesados por dos rutas, una en cada dirección.

4.1.6. Validación del sistema de enclavamientos

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 14 rutas del señalamiento original (Tabla 4.1) tienen 14 rutas equivalentes en el señalamiento generado por el RNA (Tabla 4.2), tal como se puede visualizar en la Tabla 4.3, generada automáticamente por el RNA.

Tabla 4.3: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S _{05-S₀₆}	R ₀₇	X _{15-T₀₃}
R ₀₂	S _{06-S₂₀}	R ₀₇	X _{15-T₀₃}
R ₀₃	S _{09-S₁₈}	R ₁₆	S _{27-T₀₁}
R ₀₄	S _{13-S₁₂}	R ₀₆	J _{13-C₂₅}
R ₀₅	S _{16-S₀₂}	R ₀₅	J _{12-C₂₁}
R ₀₆	S _{07-S₁₀}	R ₁₅	S _{27-S₃₅}
R ₀₇	S _{07-S₀₉}	R ₁₆	S _{27-T₀₁}
R ₀₈	S _{10-S₁₄}	R ₂₀	S _{35-J₁₄}
R ₀₉	S _{10-S₀₂}	R ₂₁	S _{35-C₂₁}
R ₁₀	S _{01-S₁₇}	R ₁₁	S _{22-S₃₂}
R ₁₁	S _{01-S₁₉}	R ₁₃	S _{22-X₁₅}
R ₁₂	S _{01-S₀₅}	R ₁₂	S _{22-T₀₅}
R ₁₃	S _{17-S₁₅}	R ₁₈	S _{32-J₁₁}
R ₁₄	S _{17-S₁₂}	R ₁₉	S _{32-C₂₅}

Las rutas R₁, R₂, R₃, R₄, R₇, R₈, R₉, R₁₀, R₁₃, R₁₄ y R₁₆ (Tabla 4.2) generadas por el RNA que no tienen equivalencias en el señalamiento original (Tabla 4.1) se deben a que el RNA creó señales extras. Las señales T₀₁, T₀₂, T₀₃, T₀₄, T₀₅ y T₀₆ fueron creadas por el RNA para proteger los finales de vía absolutos, mientras que las señales L₀₇, L₀₈, L₀₉ y L₁₀ fueron creadas para proteger los finales de vías relativos. Estas nuevas señales constituyen nuevas rutas que permiten a las formaciones detenerse previo al final (relativo o absoluto) de la red ferroviaria, lo cual incrementa la seguridad y añade nuevas rutas en sentido contrario, mejorando la logística. Estos elementos ferroviarios no se encontraban protegidos en el señalamiento original.

El señalamiento original carecía de señales protegiendo los finales de vía absolutos y relativos. No obstante, esto no se debe a un error en el diseño original, sino a que el diseño de señalamientos en la vida real se encuentra restringido por las necesidades particulares de cada locación, operador o leyes locales. Algunos operadores priorizarán tener rutas largas, dejando largas distancias sin señalamiento electrónico. Algunas normativas pueden indicar que la protección de ciertos elementos puede ser optativa. En el caso de los finales de vía pueden utilizarse señales lumínicas intermitentes que no son controladas por el señalamiento.

En otros ejemplos el señalamiento original puede estar “incompleto”, es decir, solo se consideraron las rutas en un sentido determinado, en base al uso que se le quería dar a ese señalamiento. El RNA, en cambio, siempre generará el señalamiento completo, que abarque la totalidad de la red ferroviaria ingresada, a menos que se seleccione la opción de señalamiento parcial, que respetará el sentido único de circulación que se defina en cada vía.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la

Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

4.1.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura 4.8, el ACG determinó la cantidad de elementos ferroviarios de cada tipo, tal como puede visualizarse en el Código 4.8.

Código 4.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:11  
n_switch:5  
n_doubleSwitch:0  
n_borders:4  
n_buffers:3  
n_levelCrossings:2  
n_platforms:2  
n_scissorCrossings:0  
n_signals:23  
N : 62
```

El ACG genera, en el caso de este ejemplo, 80 archivos en formato VHDL, tal como se puede visualizar en la Figura 4.9. Podemos destacar de la Figura 4.9 al archivo *Arty_Z7-10.XDC*, que define los pines de entrada y salida de la plataforma Arty Z7 10 y Arty Z7 20. Este archivo es provisto por Xilinx para esta familia de plataformas. En caso de utilizar otra plataforma, se deberá incluir el archivo XDC correspondiente. En ambos casos, cada desarrollador debe asignar manualmente los pines a cada puerto del sistema generado por el ACG.



Figura 4.9: Archivos generador por el ACG para el ejemplo 1.

Además, podemos mencionar los archivos *my-package.VHDL* y *flipFlop.VHDL*, ambos generados por el ACG. El primero es una librería que define todos los tipos de datos utilizados por el sistema, y el segundo es un flip-flop tipo D utilizado para generar la secuencia de shift registers necesarios para adaptar el clock de entrada a los diferentes dominios de clock necesarios para el timeout de cada elemento ferroviario.

Los archivos restantes son archivos que definen los módulos de alto nivel explicados en la Sección 3.2 o la representación en VHDL de cada elemento ferroviario explicado entre la Sección 3.2.8 y la Sección 3.2.14. Por ejemplo, en base lo descrito en el Código 4.8, hay 23 señales ferroviarias y podemos visualizar en la Figura 4.9 23 archivos referidos a las señales ferroviarias: desde *railwaySignal_0* hasta *railwaySignal_22*. Un ejemplo de la complejidad y profundidad del código generado es el archivo *route_13.vhdl* que se puede visualizar en el Código 4.9.

Código 4.9: route_13.vhdl

```
-- route_13.vhdl : Automatically generated using ACG
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
library work;
--Declare the package
use work.my_package.all;
entity route_13 is
port(
    clock : in std_logic := '0';
    reset : in std_logic := '0';
    routeRequest : in hex_char;
    track_ne12 : in hex_char;
    ne12_command : out routeCommands := RELEASE;
    track_ne2 : in hex_char;
    ne2_command : out routeCommands := RELEASE;
    Sw06_state : in hex_char;
    Sw06_command : out routeCommands := RELEASE;
    C25_state : in hex_char;
    C25_command : out routeCommands := RELEASE;
    L08_state : in hex_char;
    L08_command : out routeCommands := RELEASE;
    routeExecute : out hex_char
```

```

);
end entity route_13;

architecture Behavioral of route_13 is
component flipFlop is
port(
    clock : in std_logic := '0';
    reset : in std_logic := '0';
    Q : out std_logic := '0'
);
end component flipFlop;
signal restart : std_logic := '1';
signal Q : std_logic_vector(32 downto 0) := (others => '0');
signal clock_in : std_logic_vector(32 downto 0) := (others => '0');
signal timeout : std_logic := '0';
signal routeState : routeStates := WAITING_COMMAND;
signal routingIn : routeStates;
signal ne12_used : std_logic := '0';
signal ne12_state : nodeStates := FREE;
signal ne12_lock : objectLock := RELEASED;
signal ne2_state : nodeStates := FREE;
signal ne2_lock : objectLock := RELEASED;
signal Sw06_position : singleSwitchStates := NORMAL;
signal Sw06_lock : objectLock := RELEASED;
signal C25_aspectIn : signalStates := RED;
signal C25_lock: objectLock := RELEASED;
signal L08_aspectIn : signalStates := RED;
signal L08_lock : objectLock := RELEASED;
begin
    begin
        clock_in(0) <= clock;
        routingIn <= routeStates'val(to_integer(unsigned(hex_to_slv(routeRequest)))); 
        routeExecute <= slv_to_hex(std_logic_vector(to_unsigned(routeStates'pos(routeState),4)));
        ne12_state <= nodeStates'val(to_integer(unsigned(hex_to_slv(track_ne12)(2 to 3))));
        ne12_lock <= objectLock'val(to_integer(unsigned(hex_to_slv(track_ne12)(0 to 1))));
        ne2_state <= nodeStates'val(to_integer(unsigned(hex_to_slv(track_ne2)(2 to 3))));
        ne2_lock <= objectLock'val(to_integer(unsigned(hex_to_slv(track_ne2)(0 to 1))));
        Sw06_position <= singleSwitchStates'val(to_integer(unsigned(hex_to_slv(Sw06_state)(2 to 3))));
        Sw06_lock <= objectLock'val(to_integer(unsigned(hex_to_slv(Sw06_state)(0 to 1))));
        C25_aspectIn <= signalStates'val(to_integer(unsigned(hex_to_slv(C25_state)(2 to 3))));
        C25_lock <= objectLock'val(to_integer(unsigned(hex_to_slv(C25_state)(0 to 1))));
        L08_aspectIn <= signalStates'val(to_integer(unsigned(hex_to_slv(L08_state)(2 to 3))));
        L08_lock <= objectLock'val(to_integer(unsigned(hex_to_slv(L08_state)(0 to 1))));

        gen : for i in 0 to 31 generate
            inst: flipFlop port map(clock_in(i), restart, Q(i));
            clock_in(i+1) <= Q(i);
        end generate;

        process(clock,reset,Q,restart)
        begin
            if (reset = '1' or Q = "01101111100001000111010101111110") then
                timeout <= '1';
            end if;
            if (restart = '1') then
                timeout <= '0';
            end if;
        end process;

        process(clock)
        begin
            if (clock'Event and clock = '1') then
                case routeState is
                    when WAITING_COMMAND =>
                        if (routingIn = ROUTE_REQUEST) then

```

```

routeState <= RESERVING_TRACKS;
end if;
when RESERVING_TRACKS =>
    restart <= '0';
    if (routingIn = CANCEL_ROUTE or timeout ='1') then
        routeState <= CANCEL_ROUTE;
    end if;
    if ((ne12_lock = RELEASED and ne2_lock = RELEASED) and (ne2_state = FREE)) then
        ne12_command <= RESERVE;
        ne2_command <= RESERVE;
    end if;
    if (ne12_lock = RESERVED and ne2_lock = RESERVED)then
        restart <= '1';
        routeState <= LOCKING_TRACKS;
    end if;
when LOCKING_TRACKS =>
    restart <= '0';
    if (routingIn = CANCEL_ROUTE or timeout ='1') then
        routeState <= CANCEL_ROUTE;
    end if;
    if ((ne12_lock = RESERVED and ne2_lock = RESERVED) and (ne2_state = FREE)) then
        ne12_command <= LOCK;
        ne2_command <= LOCK;
    end if;
    if (ne12_lock = LOCKED and ne2_lock = LOCKED)then
        restart <= '1';
        routeState <= RESERVING_INFRASTRUCTURE;
    end if;
when RESERVING_INFRASTRUCTURE =>
    restart <= '0';
    if (routingIn = CANCEL_ROUTE or timeout ='1') then
        routeState <= CANCEL_ROUTE;
    end if;
    if (Sw06_lock = RELEASED) then
        Sw06_command <= RESERVE;
    end if;
    if (Sw06_lock = RESERVED)then
        restart <= '1';
        routeState <= LOCKING_INFRASTRUCTURE;
    end if;
when LOCKING_INFRASTRUCTURE =>
    restart <= '0';
    if (routingIn = CANCEL_ROUTE or timeout ='1') then
        routeState <= CANCEL_ROUTE;
    end if;
    if (Sw06_lock = RESERVED) then
        Sw06_command <= LOCK;
    end if;
    if (Sw06_lock = LOCKED)then
        ne12_used <= '0';
        ne2_used <= '0';
        restart <= '1';
        routeState <= DRIVING_SIGNAL;
    end if;
when DRIVING_SIGNAL =>
    restart <= '0';
    if (routingIn = CANCEL_ROUTE or timeout ='1') then
        routeState <= CANCEL_ROUTE;
    end if;
    if (C25_lock = RELEASED and L08_lock = RELEASED) then
        C25_command <= RESERVE;
        L08_command <= LOCK;
    end if;
    if (C25_lock = RESERVED and L08_lock = LOCKED) then

```

```

restart <= '1';
routeState <= SEQUENTIAL_RELEASE;
end if;
when SEQUENTIAL_RELEASE =>
restart <= '0';
if (routingIn = CANCEL_ROUTE or timeout ='1') then
routeState <= CANCEL_ROUTE;
end if;
--- Sequential release
if (ne12_used = '0' and ne12_state = OCCUPIED) then
ne12_used <= '1';
end if;
if (ne12_used = '1' and ne12_state = FREE) then
ne12_used <= '0';
ne12_command <= RELEASE;
end if;
---
if (ne12_lock = RELEASED and ne2_used = '0' and ne2_state = OCCUPIED) then
ne2_used <= '1';
--- Finish -> Release all
restart <= '1';
routeState <= RELEASING_INFRASTRUCTURE;
end if;
when RELEASING_INFRASTRUCTURE =>
Sw06_command <= RELEASE;
ne12_command <= RELEASE;
ne2_command <= RELEASE;
C25_command <= RELEASE;
L08_command <= RELEASE;
restart <= '1';
routeState <= WAITING_COMMAND;
when CANCEL_ROUTE =>
routeState <= RELEASING_INFRASTRUCTURE;
when others =>
routeState <= WAITING_COMMAND;
end case;
end if;
end process;
end Behavioral;
```

El Código 4.9 incluye la declaración de puertos, la creación de las variables necesarias, la conversión de tipos de datos, la conexión de flip-flops para generar un shift-register tal que se puedan utilizar como timeout con las condiciones prefijadas y la máquina de estados de la ruta, tal como se explicó en la Sección 3.2.14.

Cada ejemplo cuenta con su propia carpeta de principio a fin. Es decir, el archivo railML original, los archivos generados por el RNA y el código generado por el ACG se encuentran en carpetas individuales para cada ejemplo. Esto es una ventaja a la hora de mantener un orden pero una gran desventaja a la hora de sintetizar los proyectos en Vivado. Cada conjunto de archivos debería ser importado de manera individual, previa desvinculación de los archivos del proyecto anterior. Para solucionar este inconveniente se desarrolló el Código 4.10, que automatiza la importación y desvinculación de los archivos de cada ejemplo.

Código 4.10: script.tcl

```

set chosen 1

# Get a list of all design source files
set design_sources [get_files -of_objects [get_filesets sources_1]]
```

```

remove_files $design_sources

set base_folder_path "ROOT/GICSAFePhD/App/Layouts/Example_"
set folder_path "${base_folder_path}${chosen}/VHDL"

puts $folder_path

set files [glob -directory $folder_path *.vhd]

add_files -norecurse -scan_for_includes $files

update_compile_order -fileset sources_1
update_compile_order -fileset sources_1

synth_design -rtl -rtl_skip_mlo -name rtl_1

```

El parámetro *chosen* indica el número de ejemplo seleccionado, mientras que *base_folder_path* es la ruta absoluta de los ejemplos, cuyas carpetas deben ser nombradas como *Example_+chosen* para poder ser encontradas. El Código 4.10 puede ser importado en Vivado desde *Tools > Custom Commands > Customize Commands* como un archivo TCL (del inglés, Tool Command Language), que es el formato que define los comandos nativos de Vivado. Pueden importarse tantos archivos TCL como se deseen, uno por cada ejemplo a sintetizar. De esta manera, cada script aparecerá en la barra de acceso rápido de Vivado de forma independiente y se automatiza el proceso de sincronización de archivos.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, como puede verse en la parte izquierda de la Figura 4.10 donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Podemos destacar al módulo *network* que es instanciado 3 veces junto con el módulo *voter*, al ser una redundancia 2oo3, tal fue explicado en la Sección 3.3. Cada una de las instancias del módulo *network* contienen sus propias 62 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 62 en el Código 4.8.

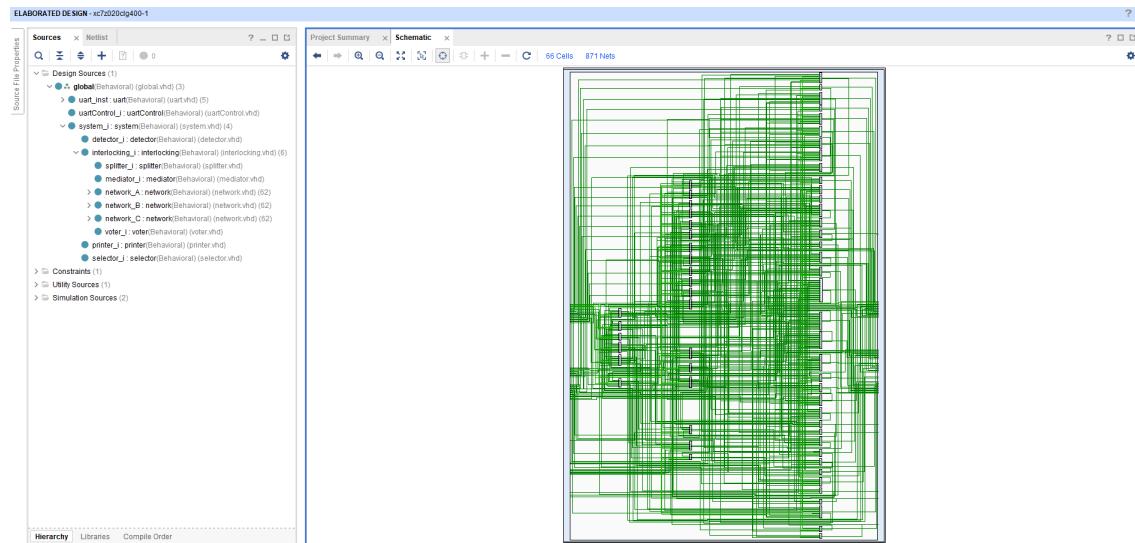


Figura 4.10: Interfaz del entorno de desarrollo Vivado para el ejemplo 1.

La parte derecha de la Figura 4.10 ilustra la representación en diagrama de bloques de uno de los módulos *network* (los tres módulos tienen los mismos bloques). Puede apreciarse en esta ventana que existen 66 módulos interconectados de forma compleja utilizando 871 señales. Pero esto es solamente una porción del sistema generado por el ACG, de inspeccionar cada uno de los bloques es posible determinar que el ejemplo 1 utiliza 9517 sub módulos conectados automáticamente mediante 21899 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Una vez que Vivado ha generado el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Durante el proceso de síntesis, Vivado busca la mejor forma de representar el código VHDL con compuertas lógicas, por lo que un código de mejor calidad brindará una representación en hardware lo mas fiel posible a lo buscado. Durante el proceso de implementación, en cambio, Vivado calcula si la representación con compuertas lógicas es posible de realizarse con la plataforma disponible. En este proceso influye no solamente la cantidad de compuertas disponibles sino también el tipo de compuertas. Si la plataforma no cuenta con la cantidad suficiente de compuertas A, entonces Vivado buscará la forma de reemplazar esa compuerta por otras compuertas B, C o D, que sean su equivalente lógico. Este proceso de reemplazo y/o simplificación puede llevar a que ambos procesos presenten discrepancias en la cantidad de elementos utilizados.

Los resultados de ambos procesos son detallados en la Tabla 4.4. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es baja y el tiempo de síntesis e implementación es de 47 y 44 segundos, respectivamente.

Tabla 4.4: Síntesis e implementación del ejemplo 1 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	3457	3416	6.42/6.50 %
FF	3810	3813	3.58 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

4.1.8. Interfaz gráfica de usuario

Una vez el bitstream generado es cargado en la FPGA, se puede visualizar la interfaz gráfica generada por el AGG (del inglés, Automatic Graphic User Interfaz Generator). El AGG genera las tramas a enviar a la FPGA y analiza las tramas de respuesta de la FPGA. Las tramas recibidas por el AGG son reinyectadas a la FPGA hasta que exista una convergencia entre las entradas y salidas de la FPGA.

El AGG se encarga, adicionalmente, de generar la interfaz gráfica de usuario utilizando la información provista por el RNA, como por ejemplo la topología (posición, conexión y largo de las vías), la distribución de la infraestructura estática (plataformas, final de vías), la disposición de la infraestructura dinámica (cambios de vías, pasos a nivel) y el señalamiento (posición y orientación de las señales).

En la Figura 4.11 se ilustra la ventana de la interfaz gráfica del ejemplo 1. La interfaz permite realizar zoom en diferentes zonas utilizando la rueda del mouse y permite desplazarse utilizando el

click izquierdo mientras se presiona la tecla shift. Ambas funcionalidades son cruciales en topologías de gran tamaño.

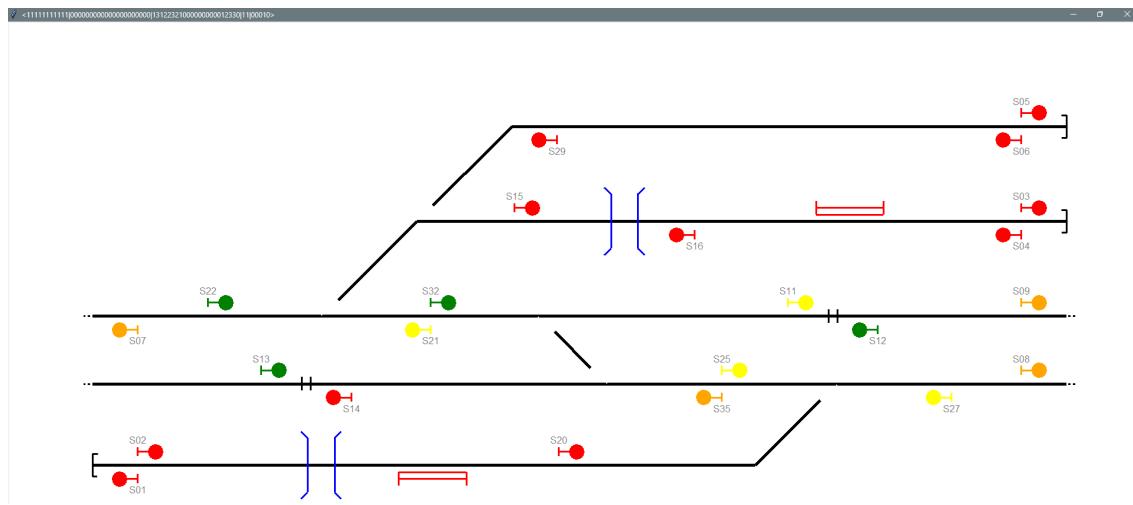


Figura 4.11: Interfaz gráfica del ejemplo 1.

El señalamiento reaccionará ante dos tipos de eventos: estáticos y dinámicos. Un evento estático involucra al usuario modificando el estado de la infraestructura, por ejemplo: ocupando o liberando una vía, cerrando o abriendo un paso a nivel o modificando la posición de un cambio de vías. Todas estas acciones se pueden realizar con un click sobre el elemento que se desea modificar. El AGG informará a la FPGA del nuevo estado del elemento seleccionado y actualizará el color y/o posición del elemento, en base a la respuesta de la FPGA. Los colores de cada elemento representan lo siguiente:

- Vías: negro (libre), rojo (ocupada), gris (enclavada).
- Paso a nivel: azul (abierto), rojo (cerrado), rojo oscuro (enclavado).
- Cambio de vías: negro (libre), gris (enclavado).

En el caso de los cambios de vías, su posición variará según la secuencia de estados posibles (por ejemplo, normal y reversa, en el caso de cambios de vías simples). El usuario es libre de elegir cualquier estado mediante clicks, pero enclavar los elementos es una función que sólo la FPGA puede ejecutar.

Por ejemplo, si se selecciona el *netElement* ne22, el AGG reportará el cambio de estado a la FPGA y esa sección cambiará su color a rojo. En pocos segundos la FPGA responderá con el nuevo estado del señalamiento, ilustrado en la Figura 4.12. La señal S32 cambiará a rojo para impedir que una formación ingrese a la sección ocupada y la señal S22 cambiará a naranja, aplicando el doble recubrimiento (ver Sección 3.1.5). El resto del señalamiento no se modificará ya que la situación local de cada señal no se modificó. Por ejemplo, la señal S29 seguirá de color rojo, para evitar un descarrilamiento, porque el trayecto que tiene a continuación se encuentra desconectado del resto de la red, porque el cambio de vías Sw04 se encuentra en posición reversa.

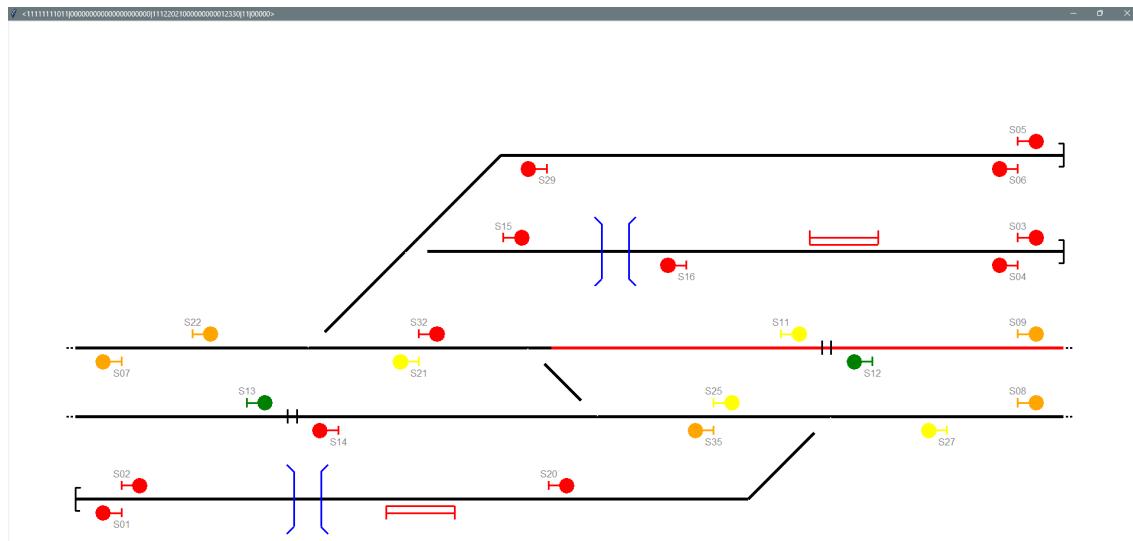


Figura 4.12: Nuevo señalamiento con ne22 ocupado.

Si en lugar de ocupar ne22 se selecciona el cambio Sw12, para modificar su posición a reversa, se obtiene la Figura 4.13. En este caso, el sistema de enclavamiento cambia ambas señales S32 y S22 a rojo, para evitar un descarrilamiento producto de que el cambio Sw12 se encuentra en posición reversa pero el cambio Sw13 aún se encuentra en posición normal.

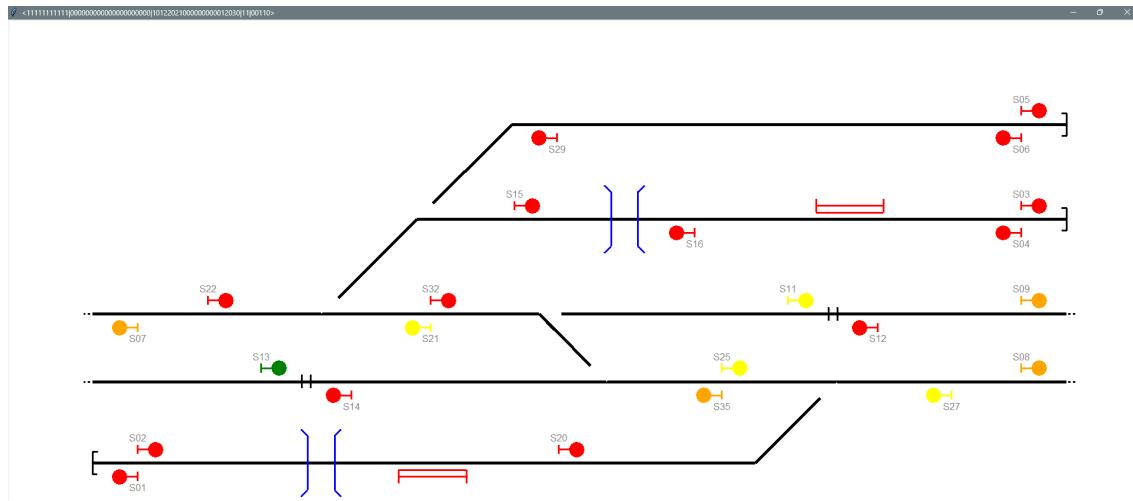


Figura 4.13: Nuevo señalamiento con Sw12 en reversa.

Un evento dinámico es aquel que se ejecuta debido a una ruta. Las rutas pueden ser solicitadas o canceladas desde la interfaz gráfica seleccionando el nombre de las señales. Cuando una señal es seleccionada, su nombre cambia a color verde y todas las señales candidatas a ser señal de finalización de ruta cambiarán su nombre a rojo. El nombre de las señales restantes cambia a color gris. Si se vuelve a seleccionar la señal de inicio, el AGG envía a la FPGA un comando de ruta cancelada para todas las rutas que inician con esa señal. Si se selecciona una de las rutas candidatas a ser señal de finalización de la ruta, el AGG envía la solicitud de esa ruta a la FPGA.

La Figura 4.14 ilustra el caso de solicitud y aprobación de la ruta R13, para lo cual el usuario seleccionó la señal S22 y luego la señal S05. Debido a que ninguna otra ruta se encontraba operativa, la FPGA puede enviar el comando de reserva y enclavamiento de las vías y la infraestructura, tal fue explicado en la Sección 3.1.2. El cambio de vías Sw04 pasa a posición reversa, el cambio de vías Sw07 pasa a posición normal y todo el trayecto queda enclavado, tal como fue explicado en la Sección 3.1.1, ilustrado en color gris. Además, la señal S22, que da inicio a la ruta, pasa a color verde y la señal S05 de finalización de ruta pasa a color rojo. Todas las señales opuestas a la ruta R13, tales como C29, pasan a color rojo.

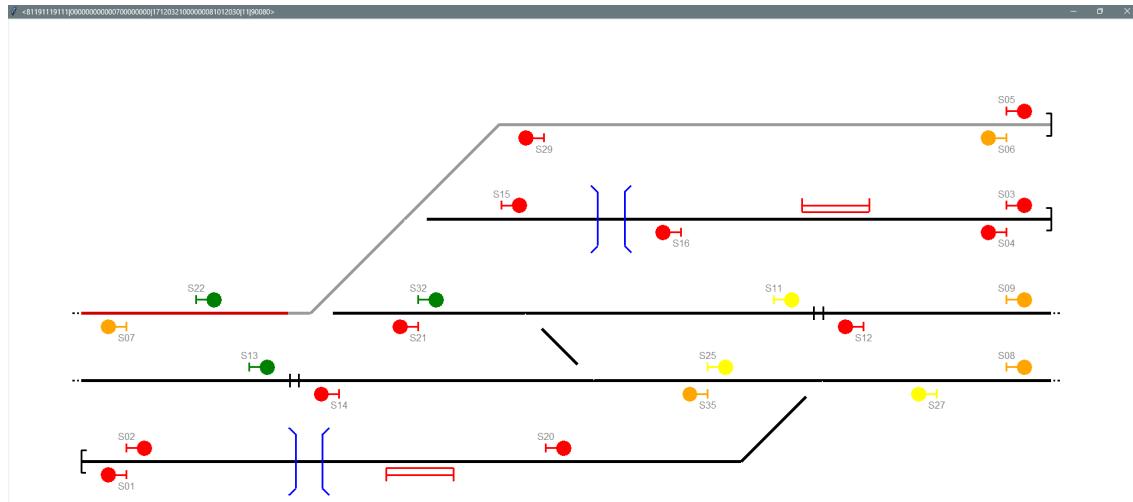


Figura 4.14: Ruta 13 del ejemplo 1 en ejecución.

Cualquier pedido de rutas que involucre la misma infraestructura que R13 no será aceptada hasta que la misma sea liberada. La liberación puede darse por tres motivos: cancelación de la ruta (ver Sección 3.1.3), timeout durante la comprobación del estado de la infraestructura (ver Sección 3.1.3) o liberación secuencial (ver Sección 3.1.6).

La Figura 4.15 ilustra el caso de solicitud y aprobación de la ruta R12, para lo cual el usuario seleccionó la señal S22 y luego la señal S15. Asumiendo que es la única ruta activada, a diferencia del caso anterior, esta ruta solicita, además de secciones libres y cambios de vías sin enclavar, que el paso a nivel Lc08 se encuentre cerrado, aún cuando el paso a nivel se encuentra después de la señal S15 de finalización de ruta. Esto se debe a la protección por solape explicada en la Sección 3.1.4.

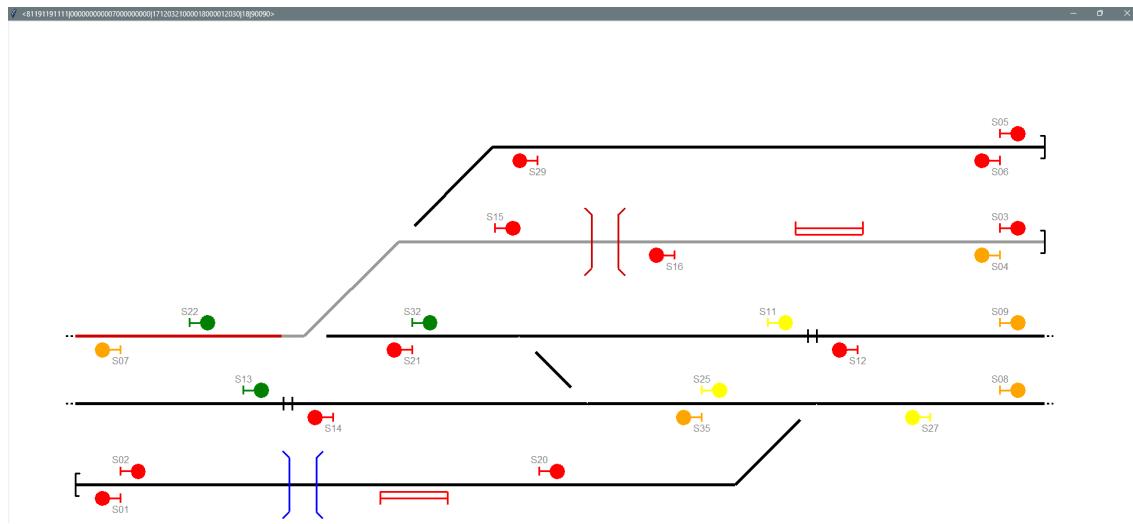


Figura 4.15: Ruta 12 del ejemplo 1 en ejecución.

La Figura 4.16 ilustra el caso de solicitud y aprobación de la ruta R19, para lo cual el usuario seleccionó la señal S32 y luego la señal S25. A diferencia de los otros casos expuestos, el sistema no sólo enclava los cambios de vías a utilizar (Sw_{12} y Sw_{13} en posición reversa), sino que también enclava el cambio de vías Sw_{06} en posición normal, debido a la protección por solape explicada en la Sección 3.1.4.

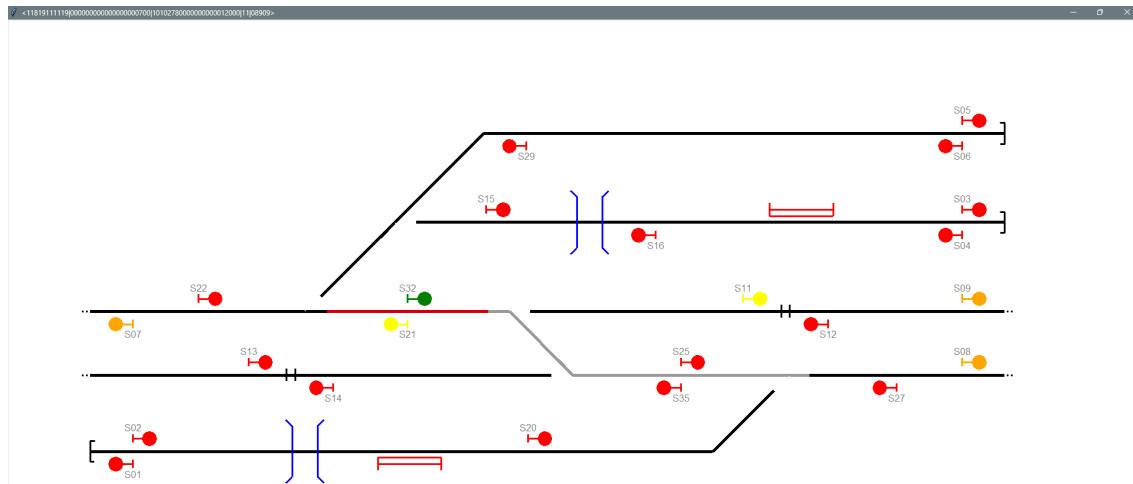


Figura 4.16: Ruta 19 del ejemplo 1 en ejecución1.

Todas las rutas fueron testeadas individualmente y en conjunto. La cancelación de rutas, timeouts y liberación secuencial fueron probadas con éxito pero al no ser posible de exponer en papel, fueron subidas en video, en un canal público para su visualización [253].

4.1.9. Comparación entre todos los ejemplos

Para una mejor comprensión de la eficacia del RNA y el ACG como herramienta para el análisis y generación de señalamiento, los resultados obtenidos en el ejemplo 1 son contrapuestos

junto a los resultados de los ejemplos 2 al 9. Estos ejemplos extra pueden encontrarse detallados en los Apéndices comprendidos entre el Apéndice A (ejemplo 2) y el Apéndice H (ejemplo 9). La comparación entre los ejemplos generados se puede visualizar en la Tabla 4.5.

Tabla 4.5: Comparación entre los ejemplos generados por el ACG.

Recursos	Ejemplo 1	Ejemplo 2	Ejemplo 3	Ejemplo 4	Ejemplo 5	Ejemplo 6	Ejemplo 7	Ejemplo 8	Ejemplo 9
netElements	11	7	53	47	8	11	7	5	7
borders	4	4	18	5	0	2	0	4	0
bufferStops	3	1	12	14	4	5	5	0	2
levelCrossings	2	1	1	2	0	0	0	6	3
platforms	2	2	13	0	0	0	0	2	5
singleSwitch	5	3	15	22	4	5	3	2	4
doubleSwitch	0	0	2	1	0	0	0	0	0
scissorSwitch	0	0	1	0	0	0	0	0	0
signals	23	12	82	77	16	24	13	16	25
routes	21	10	91	89	16	22	11	13	27
N	62	33	245	238	44	62	34	42	66
LUT	3416	1793	13509	13666	2307	3348	1798	2164	3829
FF	3813	1993	15113	15072	2347	3796	2011	2284	4137
IO	15	15	15	15	15	15	15	15	15
BUFG	3	3	3	3	3	3	3	3	3
% LUT	6,42	3,37	25,39	25,69	4,34	6,29	3,38	4,07	7,20
% FF	3,58	1,87	14,20	14,17	2,21	3,57	1,89	2,15	3,89
% IO	12,00	12,00	12,00	12,00	12,00	12,00	12,00	12,00	12,00
% BUFG	9,38	9,38	9,38	9,38	9,38	9,38	9,38	9,38	9,38

Debido a que se utiliza una interfaz serie para comunicar los datos, la cantidad de pines utilizados es independiente del tamaño del sistema a implementar, lo cual se refleja en el número de pines de entrada y salida (IOs) y de buffers (BUFGs). Sin embargo, la cantidad de Look-Up-Tables (LUTs) y Flip-Flops (FFs) incrementa considerablemente. En la Figura 4.17 se puede visualizar la cantidad de Look-Up-Table y Flip-Flops utilizadas en función de la cantidad de elementos ferroviarios (N). La cantidad de recursos utilizados incrementa linealmente, a diferencia del enfoque funcional explicado en la Sección 1.3.3, cuyo crecimiento es exponencial.

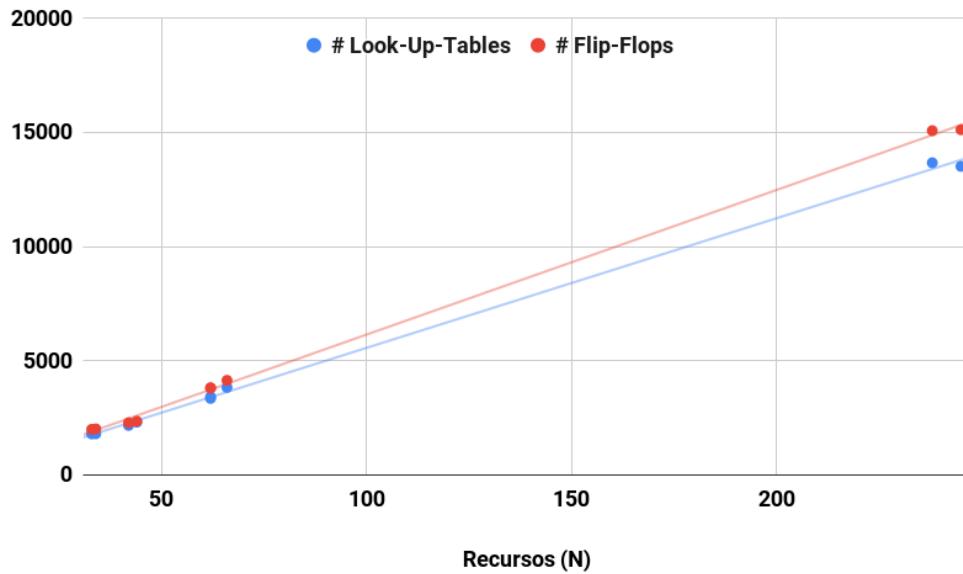


Figura 4.17: Uso de Look-Up-Tables (LUTs) y Flip-Flops (FFs) en función de N.

En la Figura 4.18 se visualizan los mismos datos de la Figura 4.17, esta vez ponderados por la cantidad de recursos disponibles en la plataforma Arty Z7 20.

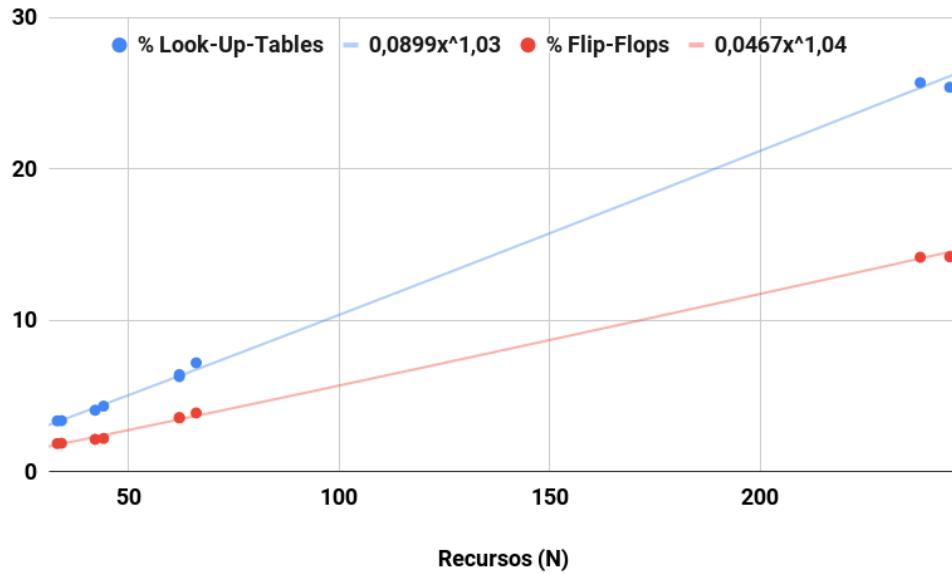


Figura 4.18: Porcentaje de uso de Look-Up-Tables (LUTs) y Flip-Flops (FFs) en función de N, considerando la plataforma Arty Z7 20.

Realizando una aproximación lineal podemos calcular el valor de N para el cual el uso de la

plataforma alcanza el 100 %. Si consideramos la cantidad de Flip-Flops, un valor de N igual a 1684 sería suficiente para que el proceso de implementación fallase por falta de recursos disponibles. Si consideramos la cantidad de Look-Up-Tables, el valor límite de N es 937. Por lo tanto, siendo que el ejemplo mas grande implementado es, según la Tabla 4.5, el ejemplo 3 con un valor de N de 245, se podría implementar un sistema hasta 3,8 veces mas grande sin inconvenientes.

Si consideramos un sistema 3,8 veces mas grande que el ejemplo 3 estaríamos hablando de una red ferroviaria de 49 estaciones, 68 cambios de vías, 193 secciones de vías, 311 semáforos y hasta 345 rutas. A modo de referencia, la línea Roca, la mas extensa del área metropolitana de Buenos Aires, cuenta con 69 estaciones a lo largo de mas de 230 kilómetros [201]. Por lo que un hipotético ejemplo que utilice el 100 % de la plataforma cubriría el 71 % de todas las funcionalidades de dicha linea.

Capítulo 5

Conclusiones

5.1. Resultados obtenidos

La adopción de un enfoque geográfico utilizando railML 3.2 como eje central del trabajo permitió desarrollar algoritmos flexibles para procesar múltiples topologías ferroviarias sin restricciones de tipo o cantidad de elementos ferroviarios incluidos. Se desarrollo una librería que cubre las 524 de railML, lo cuál amplia el universo de topologías ferroviarias analizadas automáticamente de unos pocos elementos [REF] a la totalidad de elementos ferroviarios contemplados en el estándar railML. El diseño modular de las herramientas creadas, basado en railTopoModel, permite que sea fácil de ampliar y/o mantener en futuras versiones de railML. Adicionalmente, utilizar un estándar ferroviario ampliamente utilizado permite una gran compatibilidad con las herramientas que la industria ferroviaria utiliza actualmente, lo que abre la puerta a que la herramienta sea adoptada sin dificultad.

El analizador de redes ferroviarias (RNA) genera el señalamiento apropiado, en línea con los principios de señalamiento ferroviarios definidos, para topologías ferroviarias de considerable complejidad. La tabla de enclavamientos generada es indistinguible de la original dado los parámetros de configuración del RNA correctos. En algunos casos, el RNA añade señales de protección extra para los elementos ferroviarios que fueron ignorados en el diseño original, incrementando la seguridad y mejorando la logística de la red.

La efectividad del RNA fue confirmada junto con autoridades de Trenes Argentinos que compararon el señalamiento generado por el RNA para la estación Belgrano C con el señalamiento recientemente actualizado para dicha estación. Este logro se destaca por sobre otros sistemas de generación de señalamiento que solo permiten analizar redes ferroviarias de baja complejidad, con una cantidad limitada de elementos.

El generador automático de código (ACG) permite una generación modular y escalable de cualquier locación analizada por el RNA. Incluso los ejemplos mas complejos y extensos de railML.org (ejemplo 3 y ejemplo 4) no llegan a utilizar ni un tercio de los recursos de una FPGA de bajo coste y tamaño medio. Todos los sistemas generados cuentan con: UART adaptativa, detección de tramas, triple redundancia, control de estado de enclavamientos y descentralización de las decisiones en cada elemento ferroviario.

La implementación del sistema se realiza de forma tal que solo se instancian los puertos y módulos que el RNA detectó previamente como existentes, ahorrando una gran cantidad de recursos. Adicionalmente, la implementación del comportamiento dinámico del enclavamiento basado en redes de Petri permite que se implementen la totalidad de las funcionalidades esperables en el control de rutas: pedido de rutas, cancelación de rutas, protección por aproximación, protección por solape, cancelación por timeout, protección de rutas opuestas y concatenación de rutas.

El generador automático de interfaz gráfica de usuario (AGG) permite contar con una interfaz 100 % basada en railML. El entorno gráfico es intuitivo, a semejanza de una consola real que utilizaría un ingeniero de señalamiento en su día a día. La misma es desplazable y permite el uso de zoom. El usuario puede interactuar con todos los elementos visibles en la interfaz: Las vías pueden ser ocupadas/desocupadas, los cambios de vías pueden modificar su posición, las barreras ferroviarias pueden cerrarse/abrirse y las rutas pueden solicitarse/cancelarse presionando las señales correspondientes. Cada interacción modifica en tiempo real la trama a ser enviada a la FPGA, cuya respuesta es leída y actualiza la interfaz gráfica con mínimas demoras, logrando una experiencia fluida.

Finalmente, el uso combinado de todas las herramientas creadas (RNA, ACG y AGG) permite la generación, validación e implementación en pocos segundos del señalamiento ferroviario en la totalidad de los ejemplos analizados, tanto oficiales del estándar railML como los creados en base a estaciones reales de Argentina. El análisis de la red ferroviaria cuenta con diversos parámetros, ajustables por el usuario, que permiten adaptar la herramienta a las normativas de diseño de cada país. Adicionalmente, la validación de cada regla de señalamiento ferroviario permite un diseño robusto y seguro. El código VHDL generado es compatible con cualquier familia de FPGAs y puede ser sintetizado en menos de cinco minutos. Esto reduce el tiempo de diseño e implementación del orden de meses o semanas a pocos minutos. La interfaz gráfica generada a medida de cada locación es ideal tanto para realizar pruebas dinámicas sobre la plataforma real, como también para controlar un sistema ferroviario real.

5.2. Trabajo futuro

A futuro se buscará reforzar la verificación y validación de las herramientas por medio de métodos formales, a cargo del Mg. Ing. Santiago Germino, cuyo trabajo sobre conversión del RNA en una herramienta que explote las características del formato XML de railML ya ha sido publicado [207]. Lo cual permitirá un mayor uso de la herramienta en entornos ferroviarios y su certificación para uso industrial.

Adicionalmente, se buscará adaptar el ACG para poder incorporar otros protocolos de comunicación mas robustos e implementar el sistema en la plataforma de hardware desarrollada por la Comisión Nacional de Energía Atómica (CONEA), actualmente en posesión del Laboratorio de Sistemas Embebidos.

Apéndice A

Ejemplo 2

A.1. Topología ferroviaria original

El segundo ejemplo, ilustrado en la Figura A.1, es una topología simple en base a una línea principal con una ramificación simple y una compuesta, utilizando el cambio de vías Sw01 y el par de cambio de vías Sw02 y Sw03. Ademas, se incluyeron el cruce de vías Lc01 y las plataformas Plat01 y Plat02. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología de elementos ferroviarios separados por ramificaciones simples.

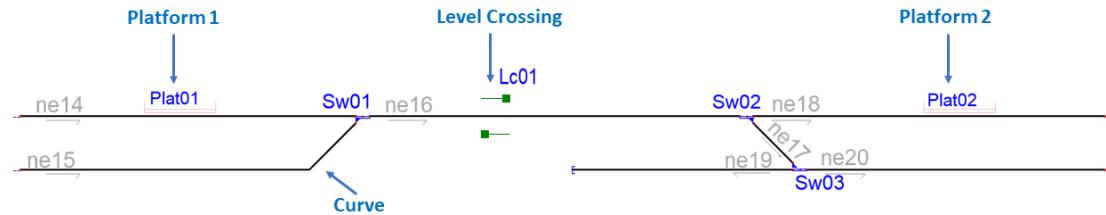


Figura A.1: Topología ferroviaria del ejemplo 2 sin señalamiento.

Para incrementar la dificultad del análisis y obtener resultados mas completos, se incluyeron curvas, finales de vías relativos y absolutos, junto con plataformas (Plat01, Plat02) y un cruce de vías (Lc01). La infraestructura se distribuyó de forma tal de tener que todos los elementos ferroviarios se encuentren aislados entre sí, separados por los cambios de vías.

A.2. Señalamiento original

El señalamiento original, ilustrado en la Figura A.2, incluye una señal de parada próxima al final de vía absoluto (S13), señales de partida en las plataformas (S07, S09), señales de protección antes de cada paso a nivel (S11, S12), una señal de maniobras antes de converger en una vía principal (S08) y señales múltiples para cambios de vías divergentes (S11, S12, S10).

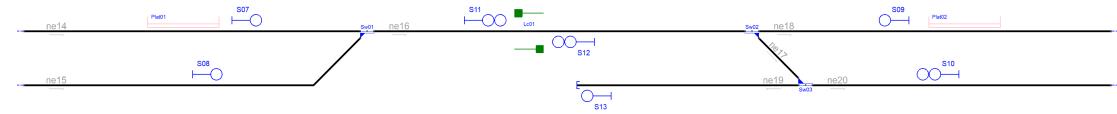


Figura A.2: Señalamiento original del ejemplo 2.

Estas señales permiten definir hasta un máximo de 5 rutas, todas ellas detalladas en la Tabla A.1. En una primera inspección, se puede comprobar que ninguno de los elementos ferroviarios son alcanzados por ninguna de las rutas. Además, todos los cambios de vías son utilizados, de forma simple o compuesta.

Tabla A.1: Tabla de enclavamiento original del ejemplo 2.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R01	S07	S11	Sw_{01}^N	-	-	ne14-ne16
R02	S08	S11	Sw_{01}^R	-	-	ne15-ne16
R03	S09	S12	Sw_{02}^N	-	-	ne18-ne16
R04	S10	S13	Sw_{03}^N	-	-	ne20-ne19
R05	S10	S12	$Sw_{03}^R + Sw_{02}^R$	-	-	ne20-ne17-ne16

Algunas rutas abarcan mas de un *netElement*, como por ejemplo la ruta R05 que comienza en la señal S10 y finaliza en la señal S12, atravesando los *netElements* ne20, ne17 y ne16, utilizando los cambios de vías Sw03 y Sw02, ambos en posición reversa.

A.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código A.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código A.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analysing railML object
Analysing graph
ne14 [-1521, 450] [-560, 450] >>
ne15 [-1521, 300] [-560, 450] >>
ne16 [-560, 450] [516, 450] >>
ne17 [516, 450] [666, 300] >>
ne18 [516, 450] [1521, 450] >>
ne19 [666, 300] [28, 300] <<
ne20 [666, 300] [1521, 300] >>
The network is connected
```

Por ejemplo, el *netElement* ne19 inicia en la coordenada (666;300) y finaliza en la coordenada (28;300). El símbolo << indica que ne19 se encuentra definido de derecha a izquierda , ya que la componente x de la coordenada final es menor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura A.2. Por ejemplo, ne14, ne15 y ne16 comparten la coordenada (-560,450), que coincide con la coordenada del cambio de vías Sw01.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código A.2.

Código A.2: Detección de puntos críticos por parte del RNA

```
Analysing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne14 has a Platform[plf116] @ [-1075, -450]
ne15 has a Curve(2 lines) @ [[-710, 300]]
ne16 has a LevelCrossing[lcr120] @ [-192, -450]
ne18 has a Platform[plf117] @ [1111, -450]
ne19 has a Middle point @ [347.0, 300]
ne20 has a Middle point @ [1093.5, 300]
```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código A.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código A.3: Infrastructure.RNA

```
Nodes: 7|Switches: 3|Signals: 0|Detectors: 0|Ends: 5|Barriers: 1
Node ne14:
    Track = track2
    Neighbours = 2 -> ['ne16', 'ne15']
Node ne15:
    Track = track3
    Neighbours = 2 -> ['ne14', 'ne16']
Node ne16:
    Track = track1
    Neighbours = 4 -> ['ne14', 'ne15', 'ne18', 'ne17']
    Level crossing -> lcr120
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [-147, -450] | Coordinate: 0.3839
    Switches -> Sw01
        ContinueCourse -> right -> ne14
        BranchCourse -> left -> ne15
    Switches -> Sw02
        ContinueCourse -> left -> ne18
        BranchCourse -> right -> ne17
Node ne17:
    Track = track5
```

```

    Neighbours = 4 -> ['ne16', 'ne18', 'ne20', 'ne19']
Node ne18:
    Track = track4
    Neighbours = 2 -> ['ne16', 'ne17']
Node ne19:
    Track = track7
    Type = BufferStop -> ['bus88']
    Neighbours = 2 -> ['ne17', 'ne20']
Node ne20:
    Track = track6
    Neighbours = 2 -> ['ne17', 'ne19']
    Switches -> Sw03
        ContinueCourse -> left -> ne19
        BranchCourse -> right -> ne17

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código ??). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código A.4: SafePoints.RNA

```

ne14:
    Next: [[-1275, -450]]
    Prev: [[-875, -450]]
ne15:
    Next: [[-810.0, 300]]
ne16:
    Next: [[-392, -450]]
    Prev: [[8, -450]]
ne18:
    Next: [[911, -450]]
    Prev: [[1311, -450]]
ne19:
    Next: [[240.7, 300], [453.3, 300]]
    Prev: [[240.7, 300], [453.3, 300]]
ne20:
    Next: [[879.8, 300], [1093.5, 300], [1307.2, 300]]
    Prev: [[879.8, 300], [1093.5, 300], [1307.2, 300]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura A.3.

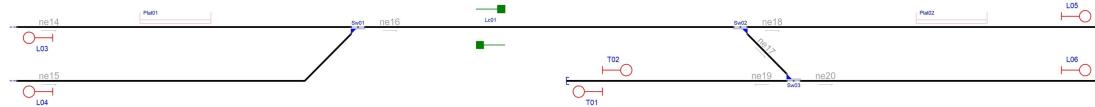


Figura A.3: Señalamiento generado por el RNA para proteger el final de vía.

Los finales de vías absolutos son protegidos por la señal de parada T01 y la señal de partida T02. A su vez, los finales de vías relativos poseen las señales de parada L03, L04, L05 y L06, cercanos al límite del externo del *netElement* al que pertenecen.

La Figura A.4 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Al aplicar el Algoritmo 8 y no detectar junturas que proteger, el RNA saltará este paso. La Figura A.4 ilustra el señalamiento sin cambios al no existir junturas que proteger.

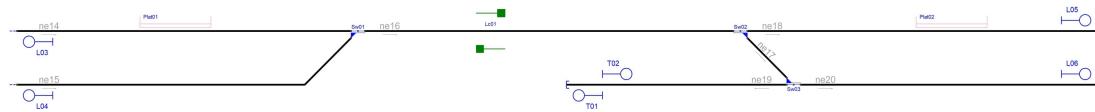


Figura A.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos. Esto no ocurre con la infraestructura de este ejemplo, ya que las plataformas Plat01 y Plat02, y el cruce de vías Lc01 se encuentran lo suficientemente espaciados entre sí. El señalamiento generado para proteger las plataformas y los cruces de vía se ilustra en rojo en la Figura A.5. Las señales generadas para proteger las plataformas son las señales de partida P09, P10, P11 y P12, mientras que las señales que protegen los cruces de vía son las señales X07 y X08.

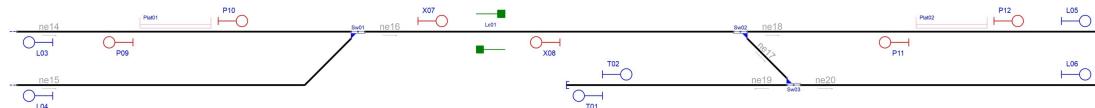


Figura A.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , se generan las señales C17, S15 y H16 para proteger el cambio de vías Sw02 y las señales C20, S21 y H22 para proteger el cambio de vías Sw03. Estas señales se encuentran resaltadas en rojo en la Figura A.6.

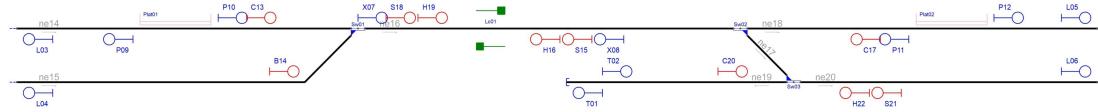


Figura A.6: Señalamiento generado por el RNA para proteger las máquinas de cambios.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías Sw02 y Sw03, desplazando las señales hasta convertirlas en las señales H16 y H22 respectivamente.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: L03, L05, X07, X08, C13, C17 y C20. La señal L03 fue eliminada por su cercanía con la señal P09, con la cual comparte dirección y sentido. Lo mismo ocurre entre la señal L05 y la señal P12; entre las señales C17 y P11; y entre las señales C20 y T02. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código A.5.

Código A.5: Reducción de señalamiento por prioridad de señales

```

removing sig20 for sig02
removing sig03 for sig09
removing sig05 for sig12
removing sig07 for sig18
removing sig08 for sig15
removing sig13 for sig10
removing sig17 for sig11
removing sig16 for sig08
removing sig19 for sig07
removing sig22 for sig21

```

El resultado de la simplificación del señalamiento se ilustra en la Figura A.7.

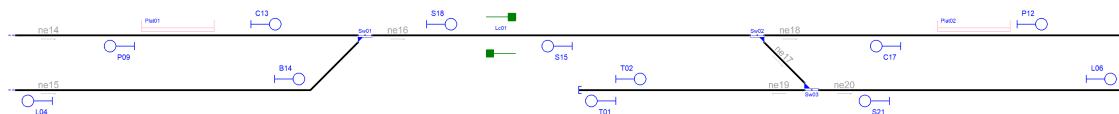


Figura A.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código A.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código A.6: Signalling.RNA

```

sig01 [T01] <<:
  From: ne19 | To: bus88_left

```

```

    Type: Stop | Direction: normal | AtTrack: left
    Position: [128, -300] | Coordinate: 0.1567
sig02 [T02] >>:
    From: ne19 | To: ne19_right
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [128, -300] | Coordinate: 0.1567
sig04 [L04] <<:
    From: ne15 | To: line113_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-1421, -300] | Coordinate: 0.3050
sig06 [L06] >>:
    From: ne20 | To: line115_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1421, -300] | Coordinate: 0.8830
sig09 [P09] <<:
    From: ne14 | To: ne14_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-1195, 450] | Coordinate: 0.9960
sig10 [P10] >>:
    From: ne14 | To: ne14_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [-955, 450] | Coordinate: 1.1063
sig11 [P11] <<:
    From: ne18 | To: ne18_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [991, 450] | Coordinate: 1.0125
sig12 [P12] >>:
    From: ne18 | To: ne18_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1231, 450] | Coordinate: 1.1437
sig14 [B14] >>:
    From: ne15 | To: ne15_right
    Type: Manouver | Direction: normal | AtTrack: left
    Position: [-810.0, -300] | Coordinate: 0.9022
sig15 [S15] <<:
    From: ne16 | To: ne16_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [8, 450] | Coordinate: 0.9890
sig18 [S18] >>:
    From: ne16 | To: ne16_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [-392, 450] | Coordinate: 0.8508
sig21 [S21] <<:
    From: ne20 | To: ne20_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [879.8, -300] | Coordinate: 0.2500

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código A.7). La misma

detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código A.7: Routes.RNA

```

route_1 [sig02 >> sig06]:
    Path: ['ne19', 'ne20']
    Switches: ['Sw03']
route_2 [sig10 >> sig18]:
    Path: ['ne14', 'ne16']
    Switches: ['Sw01', 'Sw02']
    Platforms: ['plf116']
route_3 [sig11 << sig15]:
    Path: ['ne18', 'ne16']
    Switches: ['Sw01', 'Sw02']
    Platforms: ['plf117']
route_4 [sig14 >> sig18]:
    Path: ['ne15', 'ne16']
    Switches: ['Sw01', 'Sw02']
route_5 [sig15 << sig09]:
    Path: ['ne16', 'ne14']
    Switches: ['Sw01', 'Sw02']
    Platforms: ['plf116']
route_6 [sig15 << sig04]:
    Path: ['ne16', 'ne15']
    Switches: ['Sw01', 'Sw02']
route_7 [sig18 >> sig12]:
    Path: ['ne16', 'ne18']
    Switches: ['Sw01', 'Sw02']
    Platforms: ['plf117']
route_8 [sig18 >> sig06]:
    Path: ['ne16', 'ne17', 'ne20']
    Switches: ['Sw01', 'Sw02', 'Sw03']
route_9 [sig21 << sig15]:
    Path: ['ne20', 'ne17', 'ne16']
    Switches: ['Sw01', 'Sw02', 'Sw03']
route_10 [sig21 << sig01]:
    Path: ['ne20', 'ne19']
    Switches: ['Sw03']

```

A.4. Red de grafos generada por el RNA

La información exportada en el Código A.3 (Infrastructure.RNA) Código A.4 (SafePoint.RNA), Código A.6 (Signalling.RNA) y Código A.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura A.8.

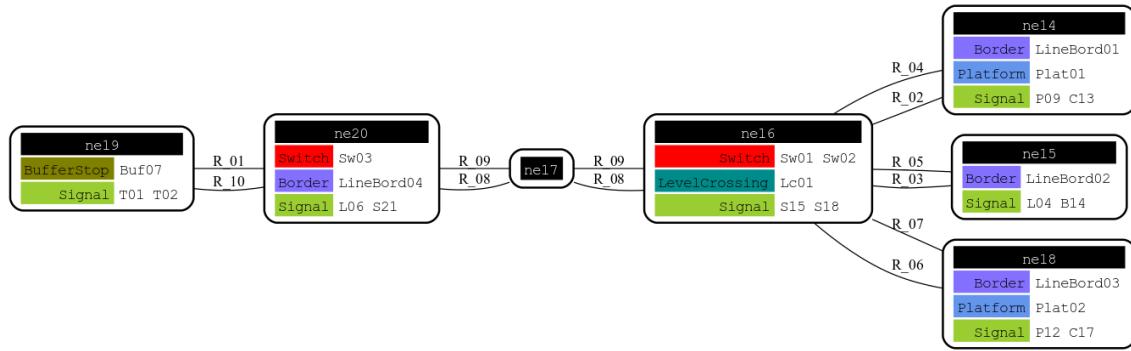


Figura A.8: Red de grafos generada por el RNA para el ejemplo 2.

Cada nodo del grafo de la Figura A.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

A.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código A.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla A.2.

Tabla A.2: Tabla de enclavamiento del ejemplo 2 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	L ₀₆	Sw ₀₃ ^N	-	-	ne ₁₉ -ne ₂₀
R ₀₂	C ₁₃	S ₁₈	Sw ₀₁ ^N	-	-	ne ₁₄ -ne ₁₆
R ₀₃	B ₁₄	S ₁₈	Sw ₀₁ ^R	-	-	ne ₁₅ -ne ₁₆
R ₀₄	S ₁₅	P ₀₉	Sw ₀₁ ^N	Plat ₀₁	Lc ₀₁	ne ₁₆ -ne ₁₄
R ₀₅	S ₁₅	L ₀₄	Sw ₀₁ ^R	Plat ₀₁	Lc ₀₁	ne ₁₆ -ne ₁₅
R ₀₆	C ₁₇	S ₁₅	Sw ₀₂ ^N	-	-	ne ₁₈ -ne ₁₆
R ₀₇	S ₁₈	P ₁₂	Sw ₀₂ ^N	Plat ₀₂	Lc ₀₁	ne ₁₆ -ne ₁₈
R ₀₈	S ₁₈	L ₀₆	Sw ₀₂ ^R +Sw ₀₃ ^R	-	Lc ₀₁	ne ₁₆ -ne ₂₀
R ₀₉	S ₂₁	S ₁₅	Sw ₀₂ ^R +Sw ₀₃ ^R	-	-	ne ₂₀ -ne ₁₆
R ₁₀	S ₂₁	T ₀₁	Sw ₀₃ ^N	-	-	ne ₂₀ -ne ₁₉

En una primera inspección podemos ver que el nuevo señalamiento tiene 10 rutas, versus las 5 rutas del señalamiento original (ver Tabla A.1). Esto se debe a que todas las vías son consideradas de ambos sentidos por el RNA, lo cuál queda de manifiesto cuando se comprueba que todas las plataformas y cruces de vía son atravesados por al menos una ruta.

A.6. Validación del sistema de enclavamientos

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 5 rutas del señalamiento original (Tabla A.1) tienen 5 rutas equivalentes en el señalamiento generado por el RNA (Tabla A.2), tal como se puede visualizar en la Tabla A.3, generada automáticamente por el RNA.

Tabla A.3: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S _{07-S₁₁}	R ₀₃	B _{14-S₁₈}
R ₀₂	S _{08-S₁₁}	R ₀₆	C _{17-S₁₅}
R ₀₃	S _{09-S₁₂}	R ₀₂	C _{13-S₁₈}
R ₀₄	S _{10-S₁₃}	R ₁₀	S _{21-S₁₅}
R ₀₅	S _{10-S₁₂}	R ₀₉	S _{21-S₁₅}

El RNA generó nuevas rutas para incrementar la seguridad de la red. La ruta R1 fue definida al crear las señales T02 y L06 para proteger el final de la vía en la sección ne19-ne20. La ruta R6 fue definida al crear la señal L04 para proteger el final de la vía en ne15, creando una nueva ruta con la señal S15. La ruta R8 fue definida al crear la señal L06 para proteger el final de la vía ne20, creando una nueva ruta con la señal S18.

Adicionalmente, el RNA generó nuevas rutas para incrementar la movilidad de la red. La ruta R5 fue definida al crear la señal de partida P09 para la plataforma plat01. La ruta R7 fue definida al crear la señal de partida P12 para la plataforma plat02.

Estos elementos ferroviarios, los finales de vía y las plataformas, no se encontraban protegidos en la tabla de enclavamientos original (Tabla A.1). Ninguna ruta original atravesaba el cruce de vías, por lo que el RNA agregó las rutas R5 y R8 que cumplen esa función.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

A.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura A.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código A.8.

Código A.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:7
n_switch:3
n_doubleSwitch:0
n_borders:4
n_buffers:1
n_levelCrossings:1
n_platforms:2
n_scissorCrossings:0
n_signals:12
N : 33
```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 51 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 2 para automatizar la importación de los archivos del ejemplo 2 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 33 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 33 en el Código A.8. El ejemplo 2 utiliza mas de 11200 sub módulos conectados automáticamente mediante mas de 23000 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla A.4. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es baja y el tiempo de síntesis e implementación es de 34 y 35 segundos, respectivamente.

Tabla A.4: Síntesis e implementación del ejemplo 2 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	1817	1793	3.42-3.37 %
FF	1990	1993	1.87 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice B

Ejemplo 3

B.1. Topología ferroviaria original

El tercer ejemplo, ilustrado en la Figura B.1, es una de los ejemplos provistos por railML.org llamado 'AdvancedExample'. Esta topología es de las mas complejas disponibles para analizar. Consiste en siete estaciones interconectadas por medio de variadas topologías ferroviarias que incluyen el uso de cambios de vías simples, dobles y en tijeras. También presenta varias playas de maniobras, curvas, pasos bajo nivel y sobre nivel. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con la topología mas compleja disponible, exigiendo al máximo al ACG y verificando la escalabilidad del sistema generado.

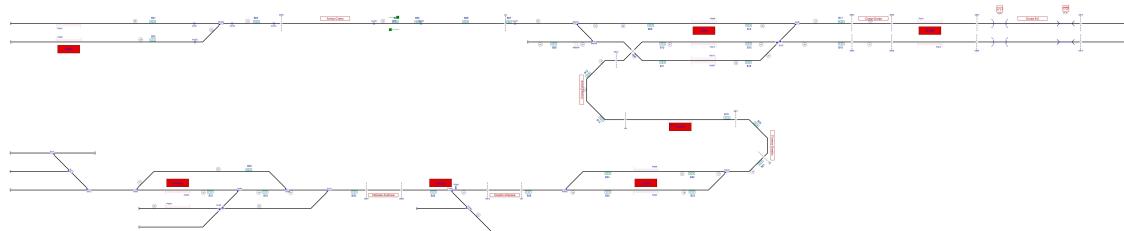


Figura B.1: Topología ferroviaria del ejemplo 3 sin señalamiento.

Debido a la alta complejidad de este ejemplo, no será posible referirse a detalles muy puntuales de la red, al ser demasiados y muy extensos. No obstante, se listarán cada una de las señales creadas en cada paso y se ilustrarán los resultados a la par del proceso. Se extraerán conclusiones generales del análisis y se dará un detalle pormenorizado de las diferencias entre el señalamiento original y el generado.

B.2. Señalamiento original

El señalamiento original, ilustrado en la Figura B.2, incluye 31 señales en total. La mayoría de las cuales se sitúan cerca de la curva y contra curva principal o próximas a las estaciones. Nombrarlas una a una sería impráctico, pero se pueden destacar a grandes rangos donde se sitúan, que funciones cumplen y cuáles descuidan. En primer lugar, todos los finales de vía absolutos se encuentran desprotegidos, así como gran parte de las playas de maniobras. En segundo lugar, el señalamiento prioriza la circulación de las formaciones en una única dirección, dependiendo de

cual vía se inspeccione. Esto lleva a concluir, a priori, que el señalamiento está subdimensionado al punto de no permitir explotar todas las características de la red ferroviaria.

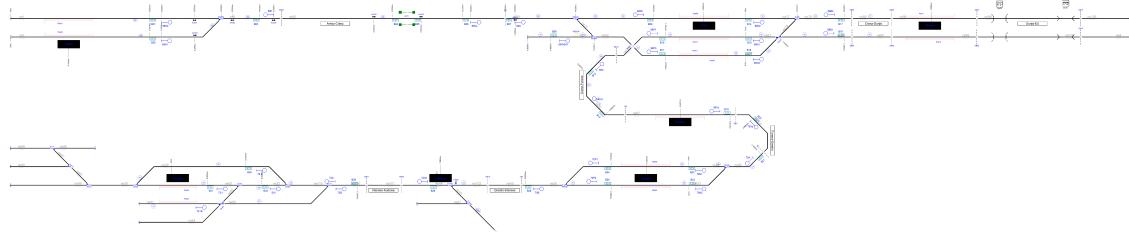


Figura B.2: Señalamiento original del ejemplo 3.

Estas señales permiten definir hasta un máximo de 33 rutas, todas ellas detalladas en la Tabla B.1. En una primera inspección, es evidente la ausencia de señalamiento en los límites de la red ferroviaria y en las playas de maniobras. Incluso pueden advertirse algunas situaciones donde las señales otorgarían autoridades ambiguas, lo cual va en contra del principio de claridad. Por ejemplo, la señal 72A podría habilitar una ruta hacia la izquierda de la topología, pero no queda claro qué camino tomaría al usar el cambio de vías Sw09, Sw04, Sw08, Sw12, Sw13 y S11; permitiendo hasta 32 posibles caminos.

Tabla B.1: Tabla de enclavamiento original del ejemplo 3.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	68N1	69Va	-	-	Lc ₀₁	ne ₇ -ne ₉₅
R ₀₂	68N2	69Va	-	-	Lc ₀₁	ne ₁ -ne ₉₅
R ₀₃	69Va	69A	-	-	-	ne ₉₅ -ne ₅₉
R ₀₄	69A	69N2	69W ₀₃ ^N	Plat ₀₉	-	ne ₅₉ -ne ₁₇
R ₀₅	69A	69N3	69W ₀₃ ^R	Plat ₁₃	-	ne ₅₉ -ne ₇₇
R ₀₆	69P2	68F	-	-	-	ne ₁₇ -ne ₉
R ₀₇	69B2	69P2	Sw ₀₆ ^N	Plat ₀₉	-	ne ₇₈ -ne ₁₇
R ₀₈	69B2	69P3	Sw ₀₆ ^R +Sw ₀₇ ^S	Plat ₁₃	-	ne ₇₈ -ne ₇₇
R ₀₉	69B2	69P1	Sw ₀₆ ^R +Sw ₀₇ ^T	Plat ₁₂	-	ne ₇₈ -ne ₂₁
R ₁₀	69C	69N1	-	Plat ₁₂	-	ne ₇₀ -ne ₂₁
R ₁₁	69Vc1	69C	-	-	-	ne ₇₀ -ne ₇₀
R ₁₂	69Vc	69Vc1	-	Plat ₀₇	-	ne ₆₇ -ne ₇₀
R ₁₃	70Va	70A	-	-	-	ne ₁₀₃ -ne ₆₄
R ₁₄	70N2	69Vc	-	-	-	ne ₂₃ -ne ₆₇
R ₁₅	70N1	69Vc	-	-	-	ne ₂₄ -ne ₆₇
R ₁₆	70P1	72Va	-	-	-	ne ₂₄ -ne ₄₄
R ₁₇	70P2	72Va	-	-	-	ne ₂₃ -ne ₄₄
R ₁₈	70B	70N2	70W ₀₂ ^N	Plat ₀₅	-	ne ₂₆ -ne ₂₃
R ₁₉	70B	70N1	70W ₀₂ ^R	Plat ₀₆	-	ne ₂₆ -ne ₂₄
R ₂₀	70A	70P1	70W ₀₁ ^N	Plat ₀₆	-	ne ₆₄ -ne ₂₄
R ₂₁	70A	70P2	70W ₀₁ ^R	Plat ₀₅	-	ne ₆₄ -ne ₂₃
R ₂₂	69W04Y	69N3	-	Plat ₁₃	-	ne ₁₄ -ne ₇₇
R ₂₃	72Va	72A	-	-	-	ne ₄₄ -ne ₁₀₀
R ₂₄	721	S01	-	-	-	ne ₈₃ -ne ₃₂
R ₂₅	723b	S01	Sw ₀₅ ^T	-	-	ne ₄₁ -ne ₃₂
R ₂₆	723b	72B	Sw ₀₅ ^S	-	-	ne ₄₁ -ne ₁₀₀
R ₂₇	722	72B	-	-	-	ne ₂₉ -ne ₁₀₀
R ₂₈	S01	72B	-	-	-	ne ₃₂ -ne ₁₀₀
R ₂₉	69B1	69P3	Sw ₀₇ ^T	Plat ₁₃	-	ne ₉₄ -ne ₇₇
R ₃₀	69B1	69P1	Sw ₀₇ ^S	Plat ₁₂	-	ne ₉₄ -ne ₂₁
R ₃₁	72B	70B	71W ₀₁ ^N	-	-	ne ₁₀₀ -ne ₂₆
R ₃₂	69P3	68F	69W ₀₄ ^R	-	-	ne ₇₇ -ne ₉
R ₃₃	69P1	70Va	-	-	-	ne ₂₁ -ne ₁₀₃

Todas las rutas abarcan mas de un *netElement*, como por ejemplo la ruta R₃₂ que comienza en la señal 69P3 y finaliza en la señal 68F, atravesando los *netElements* ne₇₇ y ne₉, utilizando el cambio de vías 69W04 en posición reversa.

B.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código B.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código B.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analysing railML object
Analysing graph
ne1 [-2010, 300] [-300, 300] >>
ne4 [7083, 150] [6686, 150] <<
ne7 [-2010, 150] [-300, 300] >>
ne9 [-300, 300] [188, 300] >>
ne11 [2580, 300] [2730, 150] >>
ne14 [2730, 150] [2190, 150] <<
ne17 [4380, 300] [2580, 300] <<
ne23 [3810, -900] [2490, -1050] <<
ne24 [3810, -900] [2490, -1050] <<
ne26 [2490, -1050] [2138, -1050] <<
ne29 [-996, -1050] [239, -1050] >>
ne30 [-996, -1050] [-1373, -1050] <<
ne32 [239, -1050] [-150, -1050] <<
ne41 [-966, -1200] [-300, -1200] >>
ne43 [1560, -1050] [1710, -1200] >>
ne44 [1560, -1050] [1165, -1050] <<
ne47 [1710, -1200] [1290, -1200] <<
ne48 [1710, -1200] [1920, -1410] >>
ne59 [2009, 300] [2580, 300] >>
ne64 [4108, -782] [3810, -900] <<
ne65 [2138, -1050] [1866, -1050] <<
ne67 [2984, -480] [3882, -480] >>
ne70 [2984, -480] [2910, 0] <<
ne78 [4380, 300] [4825, 300] >>
ne79 [4230, 150] [4380, 300] >>
ne82 [-150, -1050] [-300, -1200] <<
ne83 [-150, -1050] [-996, -1050] <<
ne84 [-300, -1200] [-966, -1350] <<
ne86 [-2013, -750] [-1673, -750] >>
ne87 [-1673, -750] [-1523, -900] >>
ne88 [-1673, -750] [-1323, -750] >>
ne89 [-1373, -1050] [-2013, -1050] <<
ne90 [-2013, -900] [-1523, -900] >>
ne91 [-1523, -900] [-1373, -1050] >>
ne93 [4825, 300] [5149, 300] >>
ne94 [4826, 150] [4230, 150] <<
ne95 [188, 300] [2009, 300] >>
ne96 [5149, 300] [5842, 300] >>
ne97 [5149, 150] [4826, 150] <<
ne98 [5842, 300] [6686, 300] >>
ne99 [5841, 150] [5149, 150] <<
ne100 [880, -1050] [578, -1050] <<
ne101 [1165, -1050] [880, -1050] <<
ne102 [1866, -1050] [1560, -1050] <<
ne103 [3882, -480] [4108, -782] >>
ne105 [6686, 300] [7083, 300] >>
ne106 [6686, 150] [5841, 150] <<
```

```

ne85 [-300, -1200] [578, -1050] >>
ne77 [4230, 150] [3045, 75] <<
ne104 [2910, 0] [3045, 75] >>
ne52 [2730, 150] [3045, 75] >>
ne21 [4230, 150] [3045, 75] <<
ne110 [578, -1050] [239, -1050] <<
The network is connected

```

Por ejemplo, el *netElement* ne110 inicia en la coordenada (578;-1050) y finaliza en la coordenada (239;-1050). El símbolo << indica que ne110 se encuentra definido de derecha a izquierda, ya que la componente x de la coordenada final es menor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura B.2. Por ejemplo, ne30, ne29 y ne83 comparten la coordenada (-996;-1050), que coincide con la coordenada del cambio de vías Sw08.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código B.2.

Código B.2: Detección de puntos críticos por parte del RNA

```

Analysing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne1 has a Platform[plf177] @ [-1539, -300]
ne4 has a Middle point @ [6884.5, 150]
ne7 has a Platform[plf178] @ [-1538, -150]
ne7 has a Curve(2 lines) @ [[-450, 150]]
ne9 has a Middle point @ [-56.0, 300]
ne14 has a Middle point @ [2460.0, 150]
ne17 has a Platform[plf185] @ [3623, -300]
ne23 has a Platform[plf181] @ [3151, 1050]
ne23 has a Curve(2 lines) @ [[3660, -1050]]
ne24 has a Platform[plf182] @ [3151, 900]
ne24 has a Curve(2 lines) @ [[2640, -900]]
ne26 has a Middle point @ [2314.0, -1050]
ne29 has a Curve(3 lines) @ [[-846, -900], [89, -900]]
ne30 has a Middle point @ [-1184.5, -1050]
ne32 has a Middle point @ [44.5, -1050]
ne41 has a Platform[plf180] @ [-653, 1200]
ne44 has a Middle point @ [1362.5, -1050]
ne47 has a Middle point @ [1500.0, -1200]
ne59 has a Middle point @ [2294.5, 300]
ne64 has a Curve(2 lines) @ [[3990, -900]]
ne65 has a Middle point @ [2002.0, -1050]
ne67 has a Platform[plf183] @ [3430, 480]
ne70 has a Curve(5 lines) @ [[2670, -150], [2820, -480], [2820, 0], [2910, 0]]
ne78 has a Middle point @ [4602.5, 300]
ne83 has a Platform[plf179] @ [-653, 1050]
ne84 has a Curve(2 lines) @ [[-450, -1350]]
ne86 has a Middle point @ [-1843.0, -750]
ne88 has a Middle point @ [-1498.0, -750]
ne89 has a Middle point @ [-1799.7, -1050]
ne89 has a Middle point @ [-1586.3, -1050]
ne90 has a Middle point @ [-1768.0, -900]
ne93 has a Middle point @ [4987.0, 300]
ne94 has a Middle point @ [4528.0, 150]
ne95 has a LevelCrossing[lcr176] @ [1100, -300]
ne96 has a Platform[plf187] @ [5459, -300]

```

```

ne97 has a Middle point @ [4987.5, 150]
ne98 has a Middle point @ [6053.0, 300]
ne98 has a Middle point @ [6264.0, 300]
ne98 has a Middle point @ [6475.0, 300]
ne99 has a Platform[plf186] @ [5459, -150]
ne100 has a Middle point @ [729.0, -1050]
ne101 has a Middle point @ [1022.5, -1050]
ne102 has a Middle point @ [1713.0, -1050]
ne103 has a Curve(4 lines) @ [[3990, -480], [4108, -782], [4140, -630]]
ne105 has a Middle point @ [6884.5, 300]
ne106 has a Middle point @ [6052.2, 150]
ne106 has a Middle point @ [6263.5, 150]
ne106 has a Middle point @ [6474.8, 150]
ne85 has a Curve(2 lines) @ [[428, -1200]]
ne77 has a Platform[plf322] @ [3619, 0]
ne77 has a Curve(3 lines) @ [[3120, 0], [4080, 0]]
ne104 has a Curve(2 lines) @ [[2970, 0]]
ne52 has a Curve(2 lines) @ [[2970, 150]]
ne21 has a Platform[plf321] @ [3621, -150]
ne21 has a Curve(2 lines) @ [[3120, 150]]
ne110 has a Middle point @ [408.5, -1050]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código B.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código B.3: Infrastructure.RNA

```

Nodes: 53|Switches: 15|Signals: 0|Detectors: 7|Ends: 30|Barriers: 1
Node ne1:
    Track = track1
    TrainDetectionElements -> ac235
        Type -> axleCounter
        Side -> right
    Type = BufferStop -> ['bus2']
    Neighbours = 2 -> ['ne9', 'ne7']
Node ne4:
    Track = track3
    Type = BufferStop -> ['bus5']
    Neighbours = 1 -> ['ne106']
Node ne7:
    Track = track5
    TrainDetectionElements -> ac236
        Type -> axleCounter
        Side -> left
    Type = BufferStop -> ['bus8']
    Neighbours = 2 -> ['ne1', 'ne9']
Node ne9:
    Track = track6
    TrainDetectionElements -> ac237
        Type -> axleCounter
        Side -> right
    TrainDetectionElements -> ac238
        Type -> axleCounter
        Side -> right
    Neighbours = 3 -> ['ne1', 'ne7', 'ne95']
    Switches -> 68W02
        ContinueCourse -> right -> ne1
        BranchCourse -> left -> ne7
Node ne11:
    Track = track8
    Neighbours = 4 -> ['ne59', 'ne17', 'ne52', 'ne14']
Node ne14:
    Track = track4

```

```

Derailer -> der172
Side -> left
Type = BufferStop -> ['bus6']
Neighbours = 2 -> ['ne11', 'ne52']
Node ne17:
    Track = track7
    Neighbours = 4 -> ['ne11', 'ne78', 'ne79', 'ne59']
Node ne23:
    Track = track12
    Neighbours = 3 -> ['ne64', 'ne24', 'ne26']
Node ne24:
    Track = track11
    Neighbours = 3 -> ['ne23', 'ne64', 'ne26']
Node ne26:
    Track = track13
    Neighbours = 3 -> ['ne23', 'ne24', 'ne65']
    Switches -> 70W02
        ContinueCourse -> right -> ne23
        BranchCourse -> left -> ne24
Node ne29:
    Track = track16
    Neighbours = 4 -> ['ne30', 'ne83', 'ne32', 'ne110']
Node ne30:
    Track = track14
    Neighbours = 4 -> ['ne29', 'ne83', 'ne89', 'ne91']
    Switches -> Sw08
        ContinueCourse -> right -> ne83
        BranchCourse -> left -> ne29
    Switches -> Sw12
        ContinueCourse -> left -> ne89
        BranchCourse -> right -> ne91
Node ne32:
    Track = track18
    Neighbours = 4 -> ['ne29', 'ne83', 'ne82', 'ne110']
    Switches -> Sw04
        ContinueCourse -> right -> ne83
        BranchCourse -> left -> ne82
Node ne41:
    Track = track19
    Type = BufferStop -> ['bus37']
    Neighbours = 3 -> ['ne82', 'ne84', 'ne85']
Node ne43:
    Track = track21
    Neighbours = 4 -> ['ne44', 'ne48', 'ne47', 'ne102']
Node ne44:
    Track = track20
    Neighbours = 3 -> ['ne43', 'ne101', 'ne102']
    Switches -> 71W01
        ContinueCourse -> left -> ne102
        BranchCourse -> right -> ne43
Node ne47:
    Track = track23
    Type = BufferStop -> ['bus50']
    Neighbours = 2 -> ['ne43', 'ne48']
Node ne48:
    Track = track22
    Neighbours = 2 -> ['ne43', 'ne47']
    Switches -> 71W02
        ContinueCourse -> right -> ne43
        BranchCourse -> left -> ne47
Node ne59:
    TrainDetectionElements -> ac241
        Type -> axleCounter
        Side -> left

```

```

Neighbours = 3 -> ['ne11', 'ne17', 'ne95']
Switches -> 69W03
    ContinueCourse -> left -> ne17
    BranchCourse -> right -> ne11
Node ne64:
    Track = track10
    Neighbours = 3 -> ['ne23', 'ne24', 'ne103']
    Switches -> 70W01
        ContinueCourse -> right -> ne24
        BranchCourse -> left -> ne23
Node ne65:
    Neighbours = 2 -> ['ne26', 'ne102']
Node ne67:
    Neighbours = 2 -> ['ne70', 'ne103']
Node ne70:
    Neighbours = 2 -> ['ne67', 'ne104']
Node ne78:
    Neighbours = 3 -> ['ne17', 'ne79', 'ne93']
    Switches -> Sw06
        ContinueCourse -> right -> ne17
        BranchCourse -> left -> ne79
Node ne79:
    Track = track24
    Neighbours = 5 -> ['ne17', 'ne78', 'ne94', 'ne77', 'ne21']
Node ne82:
    Track = track25
    Neighbours = 5 -> ['ne32', 'ne41', 'ne83', 'ne84', 'ne85']
Node ne83:
    Track = track15
    Neighbours = 4 -> ['ne29', 'ne30', 'ne32', 'ne82']
Node ne84:
    Track = track26
    Type = BufferStop -> ['bus327']
    Neighbours = 3 -> ['ne41', 'ne82', 'ne85']
Node ne86:
    Track = track28
    Type = BufferStop -> ['bus331']
    Neighbours = 2 -> ['ne88', 'ne87']
    Switches -> Sw11
        ContinueCourse -> left -> ne88
        BranchCourse -> right -> ne87
Node ne87:
    Track = track30
    Neighbours = 4 -> ['ne86', 'ne88', 'ne91', 'ne90']
Node ne88:
    Track = track29
    Type = BufferStop -> ['bus332']
    Neighbours = 2 -> ['ne86', 'ne87']
Node ne89:
    Track = track27
    Type = BufferStop -> ['bus330']
    Neighbours = 2 -> ['ne30', 'ne91']
Node ne90:
    Track = track32
    Type = BufferStop -> ['bus335']
    Neighbours = 2 -> ['ne87', 'ne91']
Node ne91:
    Track = track31
    Neighbours = 4 -> ['ne30', 'ne87', 'ne89', 'ne90']
    Switches -> Sw13
        ContinueCourse -> right -> ne87
        BranchCourse -> left -> ne90
Node ne93:
    Neighbours = 2 -> ['ne78', 'ne96']

```

```

Node ne94:
    Neighbours = 4 -> ['ne79', 'ne97', 'ne77', 'ne21']
Node ne95:
    TrainDetectionElements -> ac239
        Type -> axleCounter
        Side -> left
    TrainDetectionElements -> ac240
        Type -> axleCounter
        Side -> left
    Neighbours = 2 -> ['ne9', 'ne59']
    Level crossing -> lcr176
        Protection -> true | Barriers -> doubleHalfBarrier | Lights -> none Acoustic -> bell
        Position -> [1145, -300] | Coordinate: 0.5254
Node ne96:
    Neighbours = 2 -> ['ne93', 'ne98']
Node ne97:
    Neighbours = 2 -> ['ne94', 'ne99']
Node ne98:
    Neighbours = 2 -> ['ne96', 'ne105']
Node ne99:
    Neighbours = 2 -> ['ne97', 'ne106']
Node ne100:
    Neighbours = 3 -> ['ne101', 'ne110', 'ne85']
    Switches -> Sw41
        ContinueCourse -> right -> ne110
        BranchCourse -> left -> ne85
Node ne101:
    Neighbours = 2 -> ['ne44', 'ne100']
Node ne102:
    Neighbours = 3 -> ['ne43', 'ne44', 'ne65']
Node ne103:
    Neighbours = 2 -> ['ne64', 'ne67']
Node ne105:
    Track = track2
    Type = BufferStop -> ['bus3']
    Neighbours = 1 -> ['ne98']
Node ne106:
    Neighbours = 2 -> ['ne4', 'ne99']
Node ne85:
    Track = track33
    Neighbours = 5 -> ['ne41', 'ne82', 'ne84', 'ne100', 'ne110']
Node ne77:
    Track = track34
    Neighbours = 5 -> ['ne79', 'ne94', 'ne104', 'ne52', 'ne21']
Node ne104:
    Neighbours = 4 -> ['ne70', 'ne77', 'ne21', 'ne52']
Node ne52:
    Track = track9
    Neighbours = 5 -> ['ne11', 'ne14', 'ne77', 'ne104', 'ne21']
    Switches -> 69W04
        ContinueCourse -> left -> ne14
        BranchCourse -> right -> ne11
Node ne21:
    Track = track35
    Neighbours = 5 -> ['ne79', 'ne94', 'ne77', 'ne104', 'ne52']
Node ne110:
    Track = track17
    Neighbours = 4 -> ['ne29', 'ne32', 'ne100', 'ne85']
    Switches -> Sw09
        ContinueCourse -> left -> ne32
        BranchCourse -> right -> ne29

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de

la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código B.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código B.4: SafePoints.RNA

```

ne1:
    Next: [[-1739, -300]]
    Prev: [[-1339, -300]]
ne4:
    Next: [[6884.5, 150]]
    Prev: [[6884.5, 150]]
ne7:
    Next: [[-1738, -150], [-550.0, 150]]
    Prev: [[-1338, -150]]
ne9:
    Next: [[-56.0, 300]]
    Prev: [[-56.0, 300]]
ne14:
    Next: [[2460.0, 150]]
    Prev: [[2460.0, 150]]
ne17:
    Next: [[3423, -300]]
    Prev: [[3823, -300]]
ne23:
    Next: [[2951, 1050], [3560.0, -1050]]
    Prev: [[3351, 1050]]
ne24:
    Next: [[2951, 900]]
    Prev: [[3351, 900], [2740.0, -900]]
ne26:
    Next: [[2314.0, -1050]]
    Prev: [[2314.0, -1050]]
ne29:
    Next: [[-11.0, -900]]
    Prev: [[-746.0, -900]]
ne30:
    Next: [[-1184.5, -1050]]
    Prev: [[-1184.5, -1050]]
ne32:
    Next: [[44.5, -1050]]
    Prev: [[44.5, -1050]]
ne41:
    Next: [[-853, 1200]]
    Prev: [[-453, 1200]]
ne44:
    Next: [[1362.5, -1050]]
    Prev: [[1362.5, -1050]]
ne47:
    Next: [[1500.0, -1200]]
    Prev: [[1500.0, -1200]]
ne59:
    Next: [[2294.5, 300]]
    Prev: [[2294.5, 300]]
ne64:
    Next: [[3890.0, -900]]
ne65:
    Next: [[2002.0, -1050]]
    Prev: [[2002.0, -1050]]

```

```

ne67:
    Next: [[3230, 480]]
    Prev: [[3630, 480]]
ne70:
    Next: [[2570.0, -150], [2720.0, -480], [2810.0, 0]]
    Prev: [[2770.0, -150], [2920.0, -480], [2920.0, 0]]
ne78:
    Next: [[4602.5, 300]]
    Prev: [[4602.5, 300]]
ne83:
    Next: [[-853, 1050]]
    Prev: [[-453, 1050]]
ne84:
    Next: [[-550.0, -1350]]
ne86:
    Next: [[-1843.0, -750]]
    Prev: [[-1843.0, -750]]
ne88:
    Next: [[-1498.0, -750]]
    Prev: [[-1498.0, -750]]
ne89:
    Next: [[-1799.7, -1050], [-1586.3, -1050]]
    Prev: [[-1799.7, -1050], [-1586.3, -1050]]
ne90:
    Next: [[-1768.0, -900]]
    Prev: [[-1768.0, -900]]
ne93:
    Next: [[4987.0, 300]]
    Prev: [[4987.0, 300]]
ne94:
    Next: [[4528.0, 150]]
    Prev: [[4528.0, 150]]
ne95:
    Next: [[900, -300]]
    Prev: [[1300, -300]]
ne96:
    Next: [[5259, -300]]
    Prev: [[5659, -300]]
ne97:
    Next: [[4987.5, 150]]
    Prev: [[4987.5, 150]]
ne98:
    Next: [[6053.0, 300], [6264.0, 300], [6475.0, 300]]
    Prev: [[6053.0, 300], [6264.0, 300], [6475.0, 300]]
ne99:
    Next: [[5259, -150]]
    Prev: [[5659, -150]]
ne100:
    Next: [[729.0, -1050]]
    Prev: [[729.0, -1050]]
ne101:
    Next: [[1022.5, -1050]]
    Prev: [[1022.5, -1050]]
ne102:
    Next: [[1713.0, -1050]]
    Prev: [[1713.0, -1050]]
ne103:
    Next: [[3890.0, -480], [4008.0, -782], [4040.0, -630]]
    Prev: [[4208.0, -782], [4240.0, -630]]
ne105:
    Next: [[6884.5, 300]]
    Prev: [[6884.5, 300]]
ne106:
    Next: [[6052.2, 150], [6263.5, 150], [6474.8, 150]]

```

```

Prev: [[6052.2, 150], [6263.5, 150], [6474.8, 150]]
ne85:
  Next: [[328.0, -1200]]
ne77:
  Next: [[3419, 0], [3980.0, 0]]
  Prev: [[3819, 0], [3220.0, 0]]
ne104:
  Next: [[2870.0, 0]]
ne52:
  Next: [[2870.0, 150]]
ne21:
  Next: [[3421, -150]]
  Prev: [[3821, -150], [3220.0, 150]]
ne110:
  Next: [[408.5, -1050]]
  Prev: [[408.5, -1050]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura B.3.

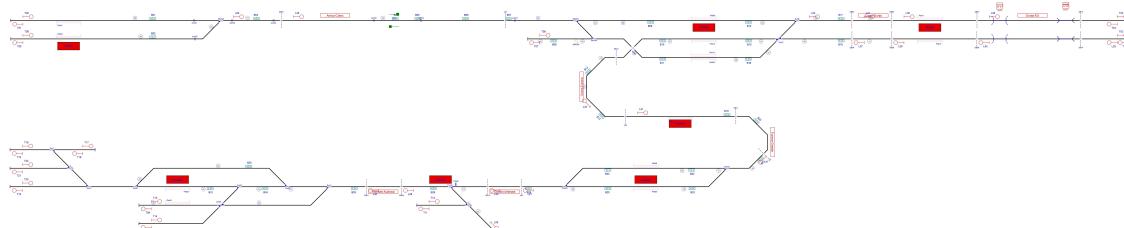


Figura B.3: Señalamiento generado por el RNA para proteger el fin de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01, T03, T05, T07, T09, T11, T13, T15, T17, T19, T21, T23 y las señales de partida son T02, T04, T06, T08, T10, T12, T14, T16, T18, T20, T22 y T24. A su vez, al no existir finales de vías relativos, el RNA no asignó señales para su protección.

La Figura B.4 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Estas señales se obtuvieron al aplicar el Algoritmo 8, tal como fue explicado en la Sección 2.2.2. Las señales generadas son todas las señales comprendidas entre J43 y J49, indicadas en color rojo. De no existir junturas que proteger, el RNA saltará este paso.

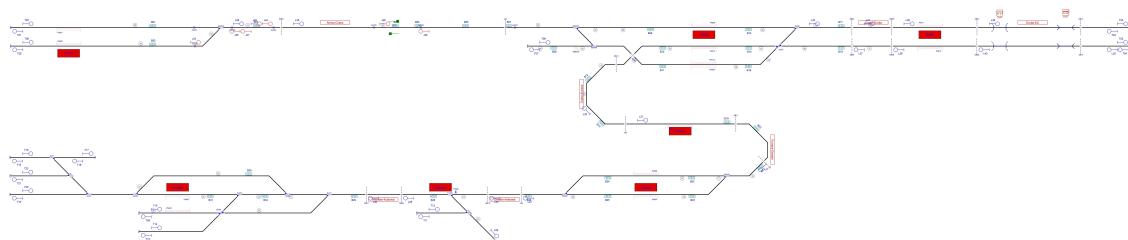


Figura B.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos. El señalamiento generado para proteger las plataformas y los cruces de vía, producto de aplicar el Algoritmo 9 y el Algoritmo 10, respectivamente, se ilustra en rojo en la Figura B.5. Las señales generadas para proteger las plataformas son las señales de partida P52 a P77, mientras que las señales que protegen los cruces de vía son las señales X50 y X51, ya que los cruces de vía Ucr01 y Ocr01 son cruces bajo nivel y sobre nivel respectivamente, por lo que no interrumpen la circulación de formaciones ferroviaria.

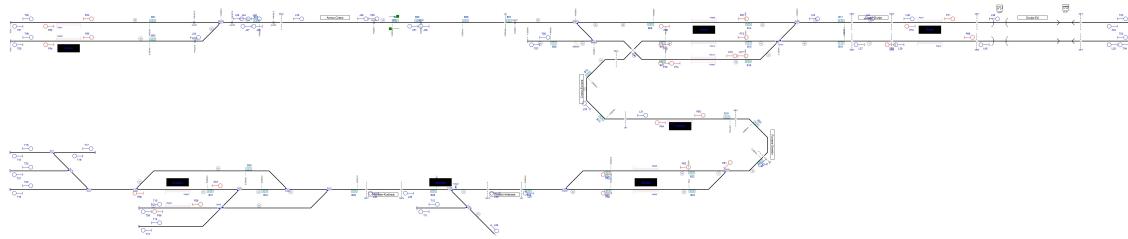


Figura B.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , se generan todas las señales resaltadas en rojo en la Figura B.6. Al tener dos cambios de vías dobles, un cambio de vías en tijeras y quince cambios de vías simples, resulta poco práctico enumerar cada una de las setenta señales generadas para proteger estos elementos ferroviarios.

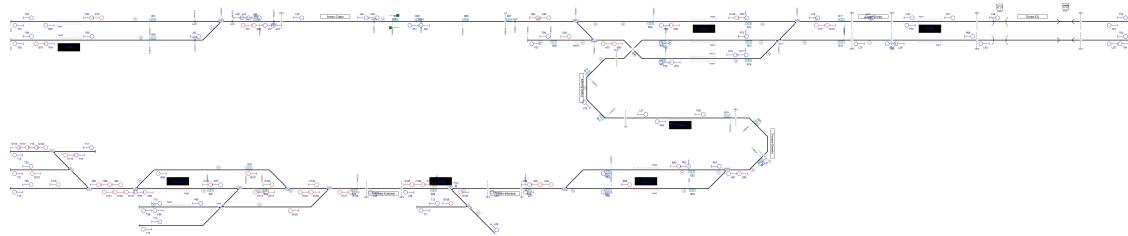


Figura B.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes,

repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías que compartan al menos una rama secundaria, desplazando las señales hasta convertirlas en las señales H en los nodos divergentes de cada cambio de vías.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son demasiadas como para ser listadas manualmente. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código B.5.

Código B.5: Reducción de señalamiento por prioridad de señales

```
Reducing redundant signals
removing sig52 for sig01
removing sig53 for sig02
removing sig25 for sig04
removing sig54 for sig05
removing sig55 for sig06
removing sig79 for sig06
removing sig85 for sig08
removing sig58 for sig09
removing sig59 for sig10
removing sig108 for sig11
removing sig116 for sig16
removing sig122 for sig16
removing sig115 for sig18
removing sig118 for sig20
removing sig123 for sig22
removing sig26 for sig44
removing sig44 for sig26
removing sig46 for sig26
removing sig71 for sig36
removing sig36 for sig71
removing sig39 for sig68
removing sig68 for sig39
removing sig47 for sig45
removing sig45 for sig80
removing sig50 for sig48
removing sig51 for sig49
removing sig78 for sig53
removing sig95 for sig56
removing sig112 for sig57
removing sig82 for sig66
removing sig109 for sig67
removing sig76 for sig74
removing sig77 for sig75
removing sig88 for sig93
removing sig107 for sig105
```

El resultado de la simplificación del señalamiento se ilustra en la Figura B.7.

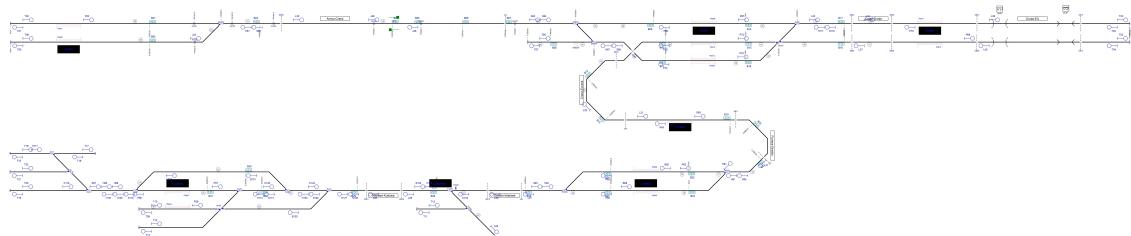


Figura B.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código B.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código B.6: Signalling.RNA

```

T01 [T01] <<:
  From: ne1 | To: bus2_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-1910, -300] | Coordinate: 0.0584
T02 [T02] >>:
  From: ne1 | To: ne1_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-1910, -300] | Coordinate: 0.0584
T03 [T03] >>:
  From: ne4 | To: bus5_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [6983, -150] | Coordinate: 0.7481
T05 [T05] <<:
  From: ne7 | To: bus8_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-1910, -150] | Coordinate: 0.1761
T06 [T06] >>:
  From: ne7 | To: ne7_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-1910, -150] | Coordinate: 0.1761
T07 [T07] <<:
  From: ne14 | To: bus6_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [2290, -150] | Coordinate: 0.1851
T08 [T08] >>:
  From: ne14 | To: ne14_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [2290, -150] | Coordinate: 0.1851
T11 [T11] <<:
  From: ne47 | To: bus50_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [1390, 1200] | Coordinate: 0.2380
T12 [T12] >>:
  From: ne47 | To: ne47_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [1390, 1200] | Coordinate: 0.2380
T13 [T13] <<:
  From: ne84 | To: bus327_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-866, 1350] | Coordinate: 0.4286
T14 [T14] >>:
  From: ne84 | To: ne84_right

```

```

Type: Stop | Direction: reverse | AtTrack: right
Position: [-866, 1350] | Coordinate: 0.4286
T15 [T15] <<:
From: ne86 | To: bus331_left
Type: Stop | Direction: reverse | AtTrack: right
Position: [-1913, 750] | Coordinate: 0.2941
T17 [T17] >>:
From: ne88 | To: bus332_right
Type: Stop | Direction: normal | AtTrack: left
Position: [-1423, 750] | Coordinate: 0.7142
T19 [T19] <<:
From: ne89 | To: bus330_left
Type: Stop | Direction: normal | AtTrack: left
Position: [-1913, 1050] | Coordinate: 0.1562
T20 [T20] >>:
From: ne89 | To: ne89_right
Type: Stop | Direction: reverse | AtTrack: right
Position: [-1913, 1050] | Coordinate: 0.1562
T21 [T21] <<:
From: ne90 | To: bus335_left
Type: Stop | Direction: reverse | AtTrack: right
Position: [-1913, 900] | Coordinate: 0.2040
T22 [T22] >>:
From: ne90 | To: ne90_right
Type: Stop | Direction: normal | AtTrack: left
Position: [-1913, 900] | Coordinate: 0.2040
T23 [T23] >>:
From: ne105 | To: bus3_right
Type: Stop | Direction: normal | AtTrack: left
Position: [6983, -300] | Coordinate: 0.7481
T24 [T24] <<:
From: ne105 | To: ne105_left
Type: Stop | Direction: reverse | AtTrack: right
Position: [6983, -300] | Coordinate: 0.7481
L25 [L25] <<:
From: ne4 | To: sb450_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [6983, -150] | Coordinate: 0.7481
L27 [L27] <<:
From: ne26 | To: sb307_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [2238, 1050] | Coordinate: 0.2840
L28 [L28] <<:
From: ne44 | To: sb437_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [1265, 1050] | Coordinate: 0.2531
L29 [L29] >>:
From: ne48 | To: line441_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [1820, 1410] | Coordinate: 0.7982
L30 [L30] <<:
From: ne65 | To: sb438_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [1966, 1050] | Coordinate: 0.3676
L32 [L32] <<:
From: ne70 | To: sb440_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [2770, 330] | Coordinate: 0.9020
L33 [L33] >>:
From: ne78 | To: sb398_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [4480, -300] | Coordinate: 0.2247
L34 [L34] >>:
From: ne93 | To: sb432_left

```

```

Type: Circulation | Direction: normal | AtTrack: left
Position: [4925, -300] | Coordinate: 0.3086
L35 [L35] >>:
From: ne95 | To: sb301_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [288, -300] | Coordinate: 0.0549
L37 [L37] <<:
From: ne97 | To: sb399_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [4926, -150] | Coordinate: 0.3095
L38 [L38] >>:
From: ne98 | To: sb449_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [5942, -300] | Coordinate: 0.1184
L40 [L40] <<:
From: ne101 | To: sb436_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [980, 1050] | Coordinate: 0.3508
L41 [L41] >>:
From: ne103 | To: sb306_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [4008, 782] | Coordinate: 0.9693
L42 [L42] <<:
From: ne106 | To: sb435_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [5941, -150] | Coordinate: 0.1183
J43 [J43] >>:
From: ne7 | To: ne7_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [-550.0, -150] | Coordinate: 0.9435
J46 [J46] >>:
From: ne9 | To: ne9_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [-56.0, -300] | Coordinate: 0.5
X50 [X50] >>:
From: ne95 | To: ne95_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [900, 300] | Coordinate: 0.5113
X51 [X51] <<:
From: ne95 | To: ne95_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [1300, 300] | Coordinate: 0.6938
P58 [P58] <<:
From: ne41 | To: ne41_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-773, -1200] | Coordinate: 3.6152
P60 [P60] <<:
From: ne23 | To: ne23_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [2871, -1050] | Coordinate: 1.6976
P63 [P63] >>:
From: ne24 | To: ne24_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [3431, -900] | Coordinate: 1.5760
P64 [P64] <<:
From: ne67 | To: ne67_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [3310, -480] | Coordinate: 1.1290
P65 [P65] >>:
From: ne67 | To: ne67_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [3550, -480] | Coordinate: 1.2410
P68 [P68] <<:
From: ne99 | To: ne99_left

```

```

Type: Circulation | Direction: normal | AtTrack: left
Position: [5179, 150] | Coordinate: 0.4356
P69 [P69] >>:
From: ne99 | To: ne99_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [5739, 150] | Coordinate: 0.9564
P70 [P70] <<:
From: ne96 | To: ne96_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [5339, 300] | Coordinate: 0.9081
P71 [P71] >>:
From: ne96 | To: ne96_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [5579, 300] | Coordinate: 1.0651
P72 [P72] <<:
From: ne21 | To: ne21_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [3341, 150] | Coordinate: 0.3936
P73 [P73] >>:
From: ne21 | To: ne21_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [3901, 150] | Coordinate: 0.7752
C78 [C78] >>:
From: ne1 | To: ne1_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [-1339, 300] | Coordinate: 0.5263
S80 [S80] <<:
From: ne9 | To: ne9_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-56.0, -300] | Coordinate: 0.5
C82 [C82] <<:
From: ne17 | To: ne17_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [3423, 300] | Coordinate: 0.5748
S83 [S83] >>:
From: ne59 | To: ne59_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [2294.5, -300] | Coordinate: 0.5
S86 [S86] <<:
From: ne52 | To: ne52_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [2870.0, -150] | Coordinate: 0.7110
B89 [B89] >>:
From: ne23 | To: ne23_right
Type: Manouver | Direction: reverse | AtTrack: right
Position: [2951, -1050] | Coordinate: 1.7090
S90 [S90] <<:
From: ne64 | To: ne64_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [3890.0, 900] | Coordinate: 0.7117
B92 [B92] >>:
From: ne24 | To: ne24_left
Type: Manouver | Direction: normal | AtTrack: right
Position: [3351, -900] | Coordinate: 1.5537
S93 [S93] >>:
From: ne26 | To: ne26_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [2314.0, 1050] | Coordinate: 0.5
C95 [C95] <<:
From: ne83 | To: ne83_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [-853, -1050] | Coordinate: 2.4880
B96 [B96] <<:
From: ne29 | To: ne29_left

```

```

Type: Manouver | Direction: normal | AtTrack: right
Position: [-746.0, 900] | Coordinate: 0.3856
S97 [S97] >>:
From: ne30 | To: ne30_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-1184.5, 1050] | Coordinate: 0.5
C100 [C100] >>:
From: ne32 | To: ne32_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [44.5, 1050] | Coordinate: 0.5
B101 [B101] <<:
From: ne29 | To: ne29_right
Type: Manouver | Direction: reverse | AtTrack: right
Position: [-11.0, 900] | Coordinate: 0.9264
S102 [S102] <<:
From: ne110 | To: ne110_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [408.5, 1050] | Coordinate: 0.5
C104 [C104] <<:
From: ne102 | To: ne102_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [1713.0, 1050] | Coordinate: 0.5
S105 [S105] >>:
From: ne44 | To: ne44_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [1362.5, 1050] | Coordinate: 0.5
C109 [C109] >>:
From: ne17 | To: ne17_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [3823, 300] | Coordinate: 0.7667
S110 [S110] <<:
From: ne78 | To: ne78_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [4602.5, -300] | Coordinate: 0.5
C114 [C114] >>:
From: ne83 | To: ne83_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-453, -1050] | Coordinate: 2.5639
S115 [S115] <<:
From: ne32 | To: ne32_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [44.5, 1050] | Coordinate: 0.5
C118 [C118] <<:
From: ne88 | To: ne88_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-1498.0, 750] | Coordinate: 0.5
S119 [S119] >>:
From: ne86 | To: ne86_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [-1843.0, 750] | Coordinate: 0.5
S122 [S122] <<:
From: ne30 | To: ne30_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [-1184.5, 1050] | Coordinate: 0.5
B130 [B130] >>:
From: ne41 | To: ne41_right
Type: Manouver | Direction: normal | AtTrack: left
Position: [-853, -1200] | Coordinate: 3.6075
S131 [S131] <<:
From: ne85 | To: ne85_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [0, 0] | Coordinate: 1.5413
B133 [B133] <<:
From: ne77 | To: ne77_right

```

```

Type: Manouver | Direction: normal | AtTrack: left
Position: [3419, 0] | Coordinate: 0.4828
S135 [S135] <<:
From: ne94 | To: ne94_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [4528.0, -150] | Coordinate: 0.5
C138 [C138] >>:
From: ne110 | To: ne110_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [408.5, 1050] | Coordinate: 0.5
S139 [S139] <<:
From: ne100 | To: ne100_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [729.0, 1050] | Coordinate: 0.5
B143 [B143] <<:
From: ne104 | To: ne104_right
Type: Manouver | Direction: reverse | AtTrack: right
Position: [2870.0, 0] | Coordinate: 1.0
S144 [S144] <<:
From: ne21 | To: ne21_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [3220.0, -150] | Coordinate: 0.1694
B145 [B145] >>:
From: ne77 | To: ne77_left
Type: Manouver | Direction: reverse | AtTrack: right
Position: [3819, 0] | Coordinate: 0.7958
S146 [S146] >>:
From: ne52 | To: ne52_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [2870.0, -150] | Coordinate: 0.7110

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código B.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código B.7: Routes.RNA

```

route_1 [T02 >> C78]:
Path: ['ne1']
Platforms: ['Plat01']
route_2 [T06 >> J43]:
Path: ['ne7']
Platforms: ['Plat02']
route_3 [T08 >> S146]:
Path: ['ne14', 'ne52']
Switches: ['69W04_N']
route_4 [T12 >> L29]:
Path: ['ne47', 'ne48']
Switches: ['71W02_R']
route_5 [T14 >> C100]:
Path: ['ne84', 'ne82', 'ne32']
Switches: ['Sw04_R', 'Sw05_RR']
route_6 [T14 >> S105]:
Path: ['ne84', 'ne85', 'ne100', 'ne101', 'ne44']
Switches: ['Sw05_NR', 'Sw41_R']
route_7 [T20 >> S97]:
Path: ['ne89', 'ne30']
Switches: ['Sw12_N']
route_8 [T22 >> S97]:
Path: ['ne90', 'ne91', 'ne30']

```

```

    Switches: ['Sw12_R', 'Sw13_R']
route_9 [T24 << P70]:
    Path: ['ne105', 'ne98', 'ne96']
    Platforms: ['Plat11']
route_10 [L25 << L42]:
    Path: ['ne4', 'ne106']
route_11 [L27 << L30]:
    Path: ['ne26', 'ne65']
    route_12 [L28 << L40]:
        Path: ['ne44', 'ne101']
route_13 [L30 << C104]:
    Path: ['ne65', 'ne102']
route_14 [L32 >> P73]:
    Path: ['ne70', 'ne104', 'ne21']
route_15 [L33 >> L34]:
    Path: ['ne78', 'ne93']
route_16 [L34 >> P71]:
    Path: ['ne93', 'ne96']
    Platforms: ['Plat11']
route_17 [L35 >> X50]:
    Path: ['ne95']
route_18 [L37 << S135]:
    Path: ['ne97', 'ne94']
route_19 [L38 >> T23]:
    Path: ['ne98', 'ne105']
route_20 [L40 << S139]:
    Path: ['ne101', 'ne100']
route_21 [L41 << S90]:
    Path: ['ne103', 'ne64']
route_22 [L42 << P68]:
    Path: ['ne106', 'ne99']
    Platforms: ['Plat10']
route_23 [J43 >> J46]:
    Path: ['ne7', 'ne9']
    Switches: ['68W02_R']
route_24 [J46 >> L35]:
    Path: ['ne9', 'ne95']
route_25 [X50 >> S83]:
    Path: ['ne95', 'ne59']
    LevelCrossings: ['Lc01']
route_26 [X51 << S80]:
    Path: ['ne95', 'ne9']
    LevelCrossings: ['Lc01']
route_27 [P60 << L27]:
    Path: ['ne23', 'ne26']
    Switches: ['70W02_N']
route_28 [P63 >> L41]:
    Path: ['ne24', 'ne64', 'ne103']
    Switches: ['70W01_N']
route_29 [P64 << L32]:
    Path: ['ne67', 'ne70']
route_30 [P65 >> L41]:
    Path: ['ne67', 'ne103']
route_31 [P68 << L37]:
    Path: ['ne99', 'ne97']
route_32 [P69 >> T03]:
    Path: ['ne99', 'ne106', 'ne4']
route_33 [P70 << S110]:
    Path: ['ne96', 'ne93', 'ne78']
route_34 [P71 >> L38]:
    Path: ['ne96', 'ne98']
route_35 [P72 << S144]:
    Path: ['ne21']
route_36 [P73 >> L33]:

```

```

Path: ['ne21', 'ne79', 'ne78']
Switches: ['Sw06_R', 'Sw07_RN']
route_37 [P73 >> P69]:
  Path: ['ne21', 'ne94', 'ne97', 'ne99']
  Switches: ['Sw07_NN']
  Platforms: ['Plat10']
route_38 [C78 >> J46]:
  Path: ['ne1', 'ne9']
  Switches: ['68W02_N']
route_39 [S80 << T01]:
  Path: ['ne9', 'ne1']
  Switches: ['68W02_N']
  Platforms: ['Plat01']
route_40 [S80 << T05]:
  Path: ['ne9', 'ne7']
  Switches: ['68W02_R']
  Platforms: ['Plat02']
route_41 [C82 << X51]:
  Path: ['ne17', 'ne59', 'ne95']
  Switches: ['69W03_N']
route_42 [S83 >> S146]:
  Path: ['ne59', 'ne11', 'ne52']
  Switches: ['69W03_R', '69W04_R']
route_43 [S83 >> C109]:
  Path: ['ne59', 'ne17']
  Switches: ['69W03_N']
  Platforms: ['Plat09']
route_44 [S86 << X51]:
  Path: ['ne52', 'ne11', 'ne59', 'ne95']
  Switches: ['69W03_R', '69W04_R']
route_45 [S86 << T07]:
  Path: ['ne52', 'ne14']
  Switches: ['69W04_N']
route_46 [B89 >> L41]:
  Path: ['ne23', 'ne64', 'ne103']
  Switches: ['70W01_R']
  Platforms: ['Plat05']
route_47 [S90 << P60]:
  Path: ['ne64', 'ne23']
  Switches: ['70W01_R']
  Platforms: ['Plat05']
route_48 [S90 << L27]:
  Path: ['ne64', 'ne23', 'ne26']
  Switches: ['70W01_R', '70W02_N']
  Platforms: ['Plat05']
route_49 [B92 >> P63]:
  Path: ['ne24']
route_50 [S93 >> B89]:
  Path: ['ne26', 'ne23']
  Switches: ['70W02_N']
route_51 [S93 >> B92]:
  Path: ['ne26', 'ne24']
  Switches: ['70W02_R']
  Platforms: ['Plat06']
route_52 [C95 << S122]:
  Path: ['ne83', 'ne30']
  Switches: ['Sw08_N']
route_53 [B96 << S122]:
  Path: ['ne29', 'ne30']
  Switches: ['Sw08_R']
route_54 [S97 >> C138]:
  Path: ['ne30', 'ne29', 'ne110']
  Switches: ['Sw08_R', 'Sw09_R']
route_55 [S97 >> C114]:

```

```

Path: ['ne30', 'ne83']
Switches: ['Sw08_N']
Platforms: ['Plat03']
route_56 [C100 >> C138]:
  Path: ['ne32', 'ne110']
  Switches: ['Sw09_N']
route_57 [B101 << B96]:
  Path: ['ne29']
route_58 [S102 << S115]:
  Path: ['ne110', 'ne32']
  Switches: ['Sw09_N']
route_59 [S102 << B101]:
  Path: ['ne110', 'ne29']
  Switches: ['Sw09_R']
route_60 [C104 << L28]:
  Path: ['ne102', 'ne44']
  Switches: ['71W01_N']
route_61 [S105 >> L29]:
  Path: ['ne44', 'ne43', 'ne48']
  Switches: ['71W01_R', '71W02_N']
route_62 [S105 >> S93]:
  Path: ['ne44', 'ne102', 'ne65', 'ne26']
  Switches: ['71W01_N']
route_63 [C109 >> L33]:
  Path: ['ne17', 'ne78']
  Switches: ['Sw06_N']
route_64 [S110 << C82]:
  Path: ['ne78', 'ne17']
  Switches: ['Sw06_N']
  Platforms: ['Plat09']
route_65 [S110 << B133]:
  Path: ['ne78', 'ne79', 'ne77']
  Switches: ['Sw06_R', 'Sw07_RR']
  Platforms: ['Plat08', 'Plat13']
route_66 [S110 << P72]:
  Path: ['ne78', 'ne79', 'ne21']
  Switches: ['Sw06_R', 'Sw07_RN']
  Platforms: ['Plat12']
route_67 [C114 >> C100]:
  Path: ['ne83', 'ne32']
  Switches: ['Sw04_N']
route_68 [S115 << C95]:
  Path: ['ne32', 'ne83']
  Switches: ['Sw04_N']
  Platforms: ['Plat03']
route_69 [S115 << P58]:
  Path: ['ne32', 'ne82', 'ne41']
  Switches: ['Sw04_R', 'Sw05_RN']
  Platforms: ['Plat04']
route_70 [S115 << T13]:
  Path: ['ne32', 'ne82', 'ne84']
  Switches: ['Sw04_R', 'Sw05_RR']
route_71 [C118 << T15]:
  Path: ['ne88', 'ne86']
  Switches: ['Sw11_N']
route_72 [S119 >> T17]:
  Path: ['ne86', 'ne88']
  Switches: ['Sw11_N']
route_73 [S119 >> S97]:
  Path: ['ne86', 'ne87', 'ne91', 'ne30']
  Switches: ['Sw11_R', 'Sw12_R', 'Sw13_N']
route_74 [S122 << T19]:
  Path: ['ne30', 'ne89']
  Switches: ['Sw12_N']

```

```

route_75 [S122 << T15]:
  Path: ['ne30', 'ne91', 'ne87', 'ne86']
  Switches: ['Sw11_R', 'Sw12_R', 'Sw13_N']
route_76 [S122 << T21]:
  Path: ['ne30', 'ne91', 'ne90']
  Switches: ['Sw12_R', 'Sw13_R']
route_77 [B130 >> C100]:
  Path: ['ne41', 'ne82', 'ne32']
  Switches: ['Sw04_R', 'Sw05_RN']
  Platforms: ['Plat04']
route_78 [B130 >> S105]:
  Path: ['ne41', 'ne85', 'ne100', 'ne101', 'ne44']
  Switches: ['Sw05_NN', 'Sw41_R']
  Platforms: ['Plat04']
route_79 [S131 << P58]:
  Path: ['ne85', 'ne41']
  Switches: ['Sw05_NN']
  Platforms: ['Plat04']
route_80 [S131 << T13]:
  Path: ['ne85', 'ne84']
  Switches: ['Sw05_NR']
route_81 [B133 << S86]:
  Path: ['ne77', 'ne52']
route_82 [S135 << B133]:
  Path: ['ne94', 'ne77']
  Switches: ['Sw07_NR']
  Platforms: ['Plat08', 'Plat13']
route_83 [S135 << P72]:
  Path: ['ne94', 'ne21']
  Switches: ['Sw07_NN']
  Platforms: ['Plat12']
route_84 [C138 >> S105]:
  Path: ['ne110', 'ne100', 'ne101', 'ne44']
  Switches: ['Sw41_N']
route_85 [S139 << S102]:
  Path: ['ne100', 'ne110']
  Switches: ['Sw41_N']
route_86 [S139 << S131]:
  Path: ['ne100', 'ne85']
  Switches: ['Sw41_R']
route_87 [B143 << L32]:
  Path: ['ne104', 'ne70']
route_88 [S144 << B143]:
  Path: ['ne21', 'ne104']
route_89 [B145 >> L33]:
  Path: ['ne77', 'ne79', 'ne78']
  Switches: ['Sw06_R', 'Sw07_RR']
route_90 [B145 >> P69]:
  Path: ['ne77', 'ne94', 'ne97', 'ne99']
  Switches: ['Sw07_NR']
  Platforms: ['Plat10']
route_91 [S146 >> B145]:
  Path: ['ne52', 'ne77']
  Platforms: ['Plat08', 'Plat13']

```

B.4. Red de grafos generada por el RNA

La información exportada en el Código B.3 (Infrastructure.RNA) Código B.4 (SafePoint.RNA), Código B.6 (Signalling.RNA) y Código B.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura

B.8.

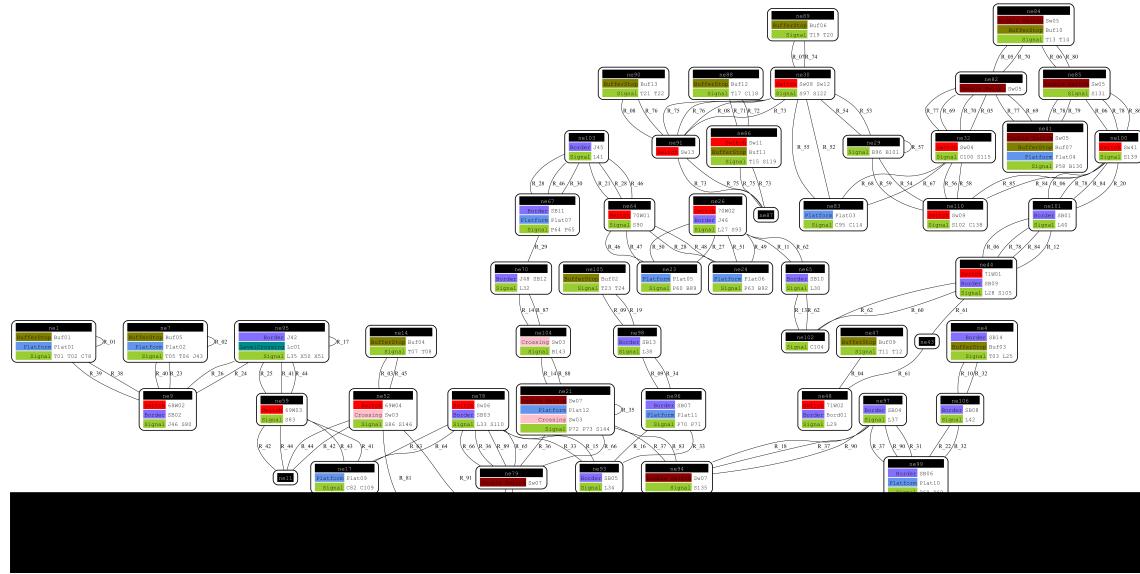


Figura B.8: Red de grafos generada por el RNA para el ejemplo 3.

Cada nodo del grafo de la Figura B.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

B.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código B.8 en una hoja de cálculo. Por una cuestión de extensión, se particionará la tabla de enclavamientos en seis tablas. Las rutas 1 a 15 se visualizan en la Tabla B.2.

Tabla B.2: Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 1 a 15).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	C ₇₈	-	Plat ₀₁	-	ne ₁ -ne ₁
R ₀₂	T ₀₆	J ₄₃	-	Plat ₀₂	-	ne ₇ -ne ₇
R ₀₃	T ₀₈	P ₁₄₆	69W ₀₄ ^N	-	-	ne ₁₄ -ne ₅₂
R ₀₄	T ₁₂	L ₂₉	71W ₀₂ ^R	-	-	ne ₄₇ -ne ₄₈
R ₀₅	T ₁₄	C ₁₀₀	Sw ₀₄ ^R	-	-	ne ₈₄ -ne ₃₂
R ₀₆	T ₁₄	S ₁₀₅	Sw ₄₁ ^R	-	-	ne ₈₄ -ne ₄₄
R ₀₇	T ₂₀	S ₉₇	Sw ₁₂ ^N	-	-	ne ₈₉ -ne ₃₀
R ₀₈	T ₂₂	S ₉₇	Sw ₁₂ ^R +Sw ₁₃ ^R	-	-	ne ₉₀ -ne ₃₀
R ₀₉	T ₂₄	P ₇₀	-	Plat ₁₁	-	ne ₁₀₅ -ne ₉₆
R ₁₀	L ₂₅	L ₄₂	-	-	-	ne ₄ -ne ₁₀₆
R ₁₁	L ₂₇	L ₃₀	-	-	-	ne ₂₆ -ne ₆₅
R ₁₂	L ₂₈	L ₄₀	-	-	-	ne ₄₄ -ne ₁₀₁
R ₁₃	L ₃₀	C ₁₀₄	-	-	-	ne ₆₅ -ne ₁₀₂
R ₁₄	L ₃₂	P ₇₃	Sw ₀₃ ^N	-	-	ne ₇₀ -ne ₂₁
R ₁₅	L ₃₃	L ₃₄	-	-	-	ne ₇₈ -ne ₉₃

Las rutas 16 a 30 se visualizan en la Tabla B.3.

Tabla B.3: Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 16 a 30).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₁₆	L ₃₄	P ₇₁	-	Plat ₁₁	-	ne ₉₃ -ne ₉₆
R ₁₇	L ₃₅	X ₅₀	-	-	-	ne ₉₅ -ne ₉₅
R ₁₈	L ₃₇	S ₁₃₅	-	-	-	ne ₉₇ -ne ₉₄
R ₁₉	L ₃₈	T ₂₃	-	-	-	ne ₉₈ -ne ₁₀₅
R ₂₀	L ₄₀	S ₁₃₉	-	-	-	ne ₁₀₁ -ne ₁₀₀
R ₂₁	L ₄₁	S ₉₀	-	-	-	ne ₁₀₃ -ne ₆₄
R ₂₂	L ₄₂	P ₆₈	-	Plat ₁₀	-	ne ₁₀₆ -ne ₉₉
R ₂₃	J ₄₃	J ₄₆	68W ₀₂ ^R	-	-	ne ₇ -ne ₉
R ₂₄	J ₄₆	L ₃₅	-	-	-	ne ₉ -ne ₉₅
R ₂₅	X ₅₀	S ₈₃	-	-	Lc ₀₁	ne ₉₅ -ne ₅₉
R ₂₆	X ₅₁	S ₈₀	-	-	Lc ₀₁	ne ₉₅ -ne ₉
R ₂₇	P ₆₀	L ₂₇	70W ₀₂ ^N	-	-	ne ₂₃ -ne ₂₆
R ₂₈	P ₆₃	L ₄₁	70W ₀₁ ^N	-	-	ne ₂₄ -ne ₁₀₃
R ₂₉	P ₆₄	L ₃₂	-	-	-	ne ₆₇ -ne ₇₀
R ₃₀	P ₆₅	L ₄₁	-	-	-	ne ₆₇ -ne ₁₀₃

Las rutas 31 a 45 se visualizan en la Tabla B.4.

Tabla B.4: Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 31 a 45).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₃₁	P ₆₈	L ₃₇	-	-	-	ne99-ne97
R ₃₂	P ₆₉	T ₀₃	-	-	-	ne99-ne4
R ₃₃	P ₇₀	S ₁₁₀	-	-	-	ne96-ne78
R ₃₄	P ₇₁	L ₃₈	-	-	-	ne96-ne98
R ₃₅	P ₇₂	S ₁₄₄	-	-	-	ne21-ne21
R ₃₆	P ₇₃	L ₃₃	Sw ₀₆ ^R	-	-	ne21-ne78
R ₃₇	P ₇₃	P ₆₉	-	Plat ₁₀	-	ne21-ne99
R ₃₈	C ₇₈	J ₄₆	68W ₀₂ ^N	-	-	ne1-ne9
R ₃₉	S ₈₀	T ₀₁	68W ₀₂ ^N	Plat ₀₁	-	ne9-ne1
R ₄₀	S ₈₀	T ₀₅	68W ₀₂ ^R	Plat ₀₂	-	ne9-ne7
R ₄₁	S ₈₂	X ₅₁	69W ₀₃ ^N	-	-	ne17-ne95
R ₄₂	S ₈₃	S ₁₄₆	69W ₀₃ ^R +69W ₀₄ ^R	-	-	ne59-ne52
R ₄₃	S ₈₃	S ₁₀₉	69W ₀₃ ^N	Plat ₀₉	-	ne59-ne17
R ₄₄	S ₈₆	X ₅₁	69W ₀₃ ^R +69W ₀₄ ^R	-	-	ne52-ne95
R ₄₅	S ₈₆	T ₀₇	69W ₀₄ ^N	-	-	ne52-ne14

Las rutas 46 a 60 se visualizan en la Tabla B.5.

Tabla B.5: Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 46 a 60).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₄₆	B ₈₉	L ₄₁	70W ₀₁ ^R	Plat ₀₅	-	ne23-ne103
R ₄₇	S ₉₀	P ₆₀	70W ₀₁ ^R	Plat ₀₅	-	ne64-ne23
R ₄₈	S ₉₀	L ₂₇	70W ₀₁ ^R +70W ₀₂ ^N	Plat ₀₅	-	ne64-ne26
R ₄₉	B ₉₂	P ₆₃	-	-	-	ne24-ne24
R ₅₀	S ₉₃	B ₈₉	70W ₀₂ ^N	-	-	ne26-ne23
R ₅₁	S ₉₃	P ₉₂	70W ₀₂ ^R	Plat ₀₆	-	ne26-ne24
R ₅₂	S ₉₅	S ₁₂₂	Sw ₀₈ ^N	-	-	ne83-ne30
R ₅₃	S ₉₆	S ₁₂₂	Sw ₀₈ ^R	-	-	ne29-ne30
R ₅₄	S ₉₇	C ₁₃₈	Sw ₀₈ ^R +Sw ₀₉ ^R	-	-	ne30-ne110
R ₅₅	S ₉₇	C ₁₁₄	Sw ₀₈ ^N	Plat ₀₃	-	ne30-ne83
R ₅₆	C ₁₀₀	S ₁₃₈	Sw ₀₉ ^N	-	-	ne32-ne110
R ₅₇	B ₁₀₁	S ₉₆	-	-	-	ne29-ne29
R ₅₈	S ₁₀₂	S ₁₁₅	Sw ₀₉ ^N	-	-	ne110-ne32
R ₅₉	S ₁₀₂	B ₁₀₁	Sw ₀₉ ^R	-	-	ne110-ne29
R ₆₀	C ₁₀₄	L ₂₈	71W ₀₁ ^N	-	-	ne102-ne44

Las rutas 61 a 75 se visualizan en la Tabla B.6.

Tabla B.6: Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 61 a 75).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₆₁	S ₁₀₅	L ₂₉	71W ₀₁ ^R +71W ₀₂ ^N	-	-	ne ₄₄ -ne ₄₈
R ₆₂	S ₁₀₅	S ₉₃	71W ₀₁ ^N	-	-	ne ₄₄ -ne ₂₆
R ₆₃	C ₁₀₉	L ₃₃	Sw ₀₆ ^N	-	-	ne ₁₇ -ne ₇₈
R ₆₄	S ₁₁₀	S ₈₂	Sw ₀₆ ^N	Plat ₀₉	-	ne ₇₈ -ne ₁₇
R ₆₅	S ₁₁₀	B ₁₃₃	Sw ₀₆ ^R +Sw ₀₇ ^{RR}	Plat ₀₈ +Plat ₁₃	-	ne ₇₈ -ne ₇₇
R ₆₆	S ₁₁₀	P ₇₂	Sw ₀₆ ^R +Sw ₀₇ ^{RN}	Plat ₁₂	-	ne ₇₈ -ne ₂₁
R ₆₇	C ₁₁₄	C ₁₀₀	Sw ₀₄ ^N	-	-	ne ₈₃ -ne ₃₂
R ₆₈	S ₁₁₅	C ₉₅	Sw ₀₄ ^N	Plat ₀₃	-	ne ₃₂ -ne ₈₃
R ₆₉	S ₁₁₅	P ₅₈	Sw ₀₄ ^R	Plat ₀₄	-	ne ₃₂ -ne ₄₁
R ₇₀	S ₁₁₃	T ₁₃	Sw ₀₄ ^R	-	-	ne ₃₂ -ne ₈₄
R ₇₁	S ₁₁₈	T ₁₅	Sw ₁₁ ^N	-	-	ne ₈₈ -ne ₈₆
R ₇₂	S ₁₁₉	T ₁₇	Sw ₁₁ ^N	-	-	ne ₈₆ -ne ₈₈
R ₇₃	S ₁₁₉	S ₉₇	Sw ₁₁ ^R +Sw ₁₂ ^R +Sw ₁₃ ^N	-	-	ne ₈₆ -ne ₃₀
R ₇₄	S ₁₂₂	T ₁₉	Sw ₁₂ ^N	-	-	ne ₃₀ -ne ₈₉
R ₇₅	S ₁₂₂	T ₁₅	Sw ₁₁ ^R +Sw ₁₂ ^R +Sw ₁₃ ^N	-	-	ne ₃₀ -ne ₈₆

Las rutas 75 a 91 se visualizan en la Tabla B.7.

Tabla B.7: Tabla de enclavamiento del ejemplo 3 generada por el RNA (Rutas 75 a 91).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₇₆	S ₁₂₂	T ₂₁	Sw ₁₂ ^R +Sw ₁₃ ^R	-	-	ne ₃₀ -ne ₉₀
R ₇₇	B ₁₃₀	C ₁₀₀	Sw ₀₄ ^R +Sw ₀₅ ^{RN}	Plat ₀₄	-	ne ₄₁ -ne ₃₂
R ₇₈	B ₁₃₀	S ₁₀₅	Sw ₀₅ ^{NN} +Sw ₄₁ ^R	Plat ₀₄	-	ne ₄₁ -ne ₄₄
R ₇₉	S ₁₃₁	P ₅₈	Sw ₀₅ ^{NN}	Plat ₀₄	-	ne ₈₅ -ne ₄₁
R ₈₀	S ₁₃₁	T ₁₃	Sw ₀₅ ^{NR}	-	-	ne ₈₅ -ne ₈₄
R ₈₁	B ₁₃₃	S ₈₆	-	-	-	ne ₇₇ -ne ₅₂
R ₈₂	S ₁₃₅	B ₁₃₃	Sw ₀₇ ^{NR}	Plat ₀₈ +Plat ₁₃	-	ne ₉₄ -ne ₇₇
R ₈₃	S ₁₃₅	P ₇₂	Sw ₀₇ ^{NN}	Plat ₁₂	-	ne ₉₄ -ne ₂₁
R ₈₄	C ₁₃₈	S ₁₀₅	Sw ₄₁ ^N	-	-	ne ₁₁₀ -ne ₄₄
R ₈₅	S ₁₃₉	S ₁₀₂	Sw ₄₁ ^N	-	-	ne ₁₀₀ -ne ₁₁₀
R ₈₆	S ₁₃₉	S ₁₃₁	Sw ₄₁ ^R	-	-	ne ₁₀₀ -ne ₈₅
R ₈₇	B ₁₄₃	L ₃₂	-	-	-	ne ₁₀₄ -ne ₇₀
R ₈₈	S ₁₄₄	B ₁₄₃	-	-	-	ne ₂₁ -ne ₁₀₄
R ₈₉	B ₁₄₅	L ₃₃	Sw ₀₆ ^R +Sw ₀₇ ^{RR}	-	-	ne ₇₇ -ne ₇₈
R ₉₀	B ₁₄₅	P ₆₉	Sw ₀₇ ^{NR}	Plat ₁₀	-	ne ₇₇ -ne ₉₉
R ₉₁	S ₁₄₆	B ₁₄₅	-	Plat ₀₈ +Plat ₁₃	-	ne ₅₂ -ne ₇₇

En una primera inspección podemos ver que el nuevo señalamiento tiene 91 rutas, versus las 33 rutas del señalamiento original (ver Tabla B.1). Esto se debe a la gran cantidad de secciones de la red que no cuentan con señalamiento apropiado.

B.6. Validación del sistema

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 33 rutas del señalamiento original (Tabla B.1) tienen 33 rutas equivalentes en el señalamiento generado por el RNA (Tabla B.2, B.3, B.4, B.5, B.6, B.7), tal como se puede visualizar en la Tabla B.8, generada automáticamente por el RNA.

Tabla B.8: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	68N1-69Va	R ₂₃ +R ₂₄	J ₄₃ -L ₃₅
R ₀₂	68N2-69Va	R ₃₈ +R ₂₄	C ₇₈ -L ₃₅
R ₀₃	69Va-69A	R ₂₅	X ₅₀ -S ₈₃
R ₀₄	69A-69N2	R ₄₃	S ₈₃ -S ₁₀₉
R ₀₅	69A-69N3	R ₄₂ +R ₉₁	S ₈₃ -B ₁₄₅
R ₀₆	69P2-68F	R ₄₁ +R ₂₆	S ₈₂ -S ₈₀
R ₀₇	69B2-69P2	R ₆₄	S ₁₁₀ -S ₈₂
R ₀₈	69B2-69P3	R ₆₅	S ₁₁₀ -B ₁₃₃
R ₀₉	69B2-69P1	R ₆₆	S ₁₁₀ -P ₇₂
R ₁₀	69C-69N1	R ₁₄	L ₃₂ -T ₇₃
R ₁₁	69Vc1-69C	R ₈₈ +R ₈₇	S ₁₄₄ -L ₃₂
R ₁₂	69Vc-69Vc1	R ₂₉	P ₆₄ -L ₃₂
R ₁₃	70Va-70A	R ₂₁	L ₄₁ -S ₉₀
R ₁₄	70N2-69Vc	R ₂₇	P ₆₀ -L ₂₇
R ₁₅	70N1-69Vc	R ₂₈	P ₆₃ -L ₄₁
R ₁₆	70P1-72Va	R ₂₈ +R ₂₁ +R ₄₈ +R ₁₁ +R ₁₃ +R ₆₀	P ₆₃ -L ₂₈
R ₁₇	70P2-72Va	R ₂₇ +R ₁₁ +R ₁₃ +R ₆₀	P ₆₀ -L ₂₈
R ₁₈	70B-70N2	R ₅₀	S ₉₃ -B ₈₉
R ₁₉	70B-70N1	R ₅₁	S ₉₃ -P ₉₂
R ₂₀	70A-70P1	R ₄₈ +R ₅₁	S ₉₀ -P ₉₂
R ₂₁	70A-70P2	R ₄₇	S ₉₀ -P ₆₀
R ₂₂	69W04Y-69N3	R ₀₃ +R ₉₁	T ₀₈ -B ₁₄₅
R ₂₃	72Va-72A	R ₁₂ +R ₂₀	L ₂₈ -S ₁₃₉
R ₂₄	721-S01	R ₆₇	C ₁₁₄ -C ₁₀₀
R ₂₅	723b-S01	R ₇₇	B ₁₃₀ -C ₁₀₀
R ₂₆	723b-72B	R ₇₈ +R ₁₂ +R ₂₀	B ₁₃₀ -S ₁₃₉
R ₂₇	722-72B	R ₅₃ +R ₅₄ +R ₈₄ +R ₁₂ +R ₂₀	S ₉₆ -S ₁₂₂
R ₂₈	S01-72B	R ₇₀ +R ₀₆ +R ₁₂ +R ₂₀	S ₁₁₃ -S ₁₂₂
R ₂₉	69B1-69P3	R ₈₂	S ₁₃₅ -B ₁₃₃
R ₃₀	69B1-69P1	R ₈₃	S ₁₃₅ -P ₇₂
R ₃₁	72B-70B	R ₈₅ +R ₈₄ +R ₆₂	S ₁₃₉ -S ₉₃
R ₃₂	69P3-68F	R ₈₁ +R ₄₄ +R ₂₆	B ₁₃₃ -S ₈₀
R ₃₃	69P1-70Va	R ₃₆	P ₇₃ -L ₃₃

Las rutas R1, R2, R5, R6, R11, R16, R17, R20, R22, R23, R26, R27, R28, R31 y R32 del señalamiento original fueron divididas en rutas más pequeñas en el señalamiento generado por el RNA. La razón para dividir la ruta depende de cada caso, tal sea por su longitud y/o por abarcar diversos elementos ferroviarios. Por ejemplo, la ruta R32 fue dividida por ambos motivos: es muy extensa y atraviesa un cambio de vías en tijeras, dos cambios de vías simples, un cruce de vías y una plataforma. Las rutas producto de particionar R32 (R81, R44 y R26 del nuevo señalamiento) añaden paradas luego de cruzar el cambio de vías en tijeras y antes del cruce de vías, incrementando la seguridad y flexibilidad en la logística de la red. Un análisis similar puede hacerse con las demás rutas particionadas.

De las 91 rutas generadas por el RNA, 29 son particiones de rutas originales, por lo que 62 de ellas están relacionadas de alguna manera a las 33 rutas originales. Las 29 rutas restantes corresponden a la protección de los 12 finales de vía absolutos y al único final de vía relativo, además de las señales añadidas para poder operar las playas de maniobras que se encontraban sin señalamiento.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

B.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura B.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código B.8.

Código B.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:53
n_switch:15
n_doubleSwitch:2
n_borders:18
n_buffers:12
n_levelCrossings:1
n_platforms:13
n_scissorCrossings:1
n_signals:82
N : 245
```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 253 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el

parámetro *chosen* a 3 para automatizar la importación de los archivos del ejemplo 3 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 245 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 245 en el Código B.8. El ejemplo 3 utiliza mas de 70600 sub módulos conectados automáticamente mediante mas de 147300 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla B.9. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es intermedia y el tiempo de síntesis e implementación es de 2 minutos con 6 segundos y 3 minuto con 39 segundos, respectivamente.

Tabla B.9: Síntesis e implementación del ejemplo 3 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	13626	13509	25.61-25.39 %
FF	15110	15113	14.20 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice C

Ejemplo 4

C.1. Topología ferroviaria original

El cuarto ejemplo , , es una topología incluida dentro de los ejemplos de railML.org llamada 'ERTMS_Test_Plant.RailAID'. La misma es una topología de tres estaciones con bypasses dobles y dos playas de maniobras, separadas entre si por largas distancias, atravesadas por un cruce de vías. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología extensa y ampliamente estudiada por la comunidad de railML. Debido a su extensión, la Figura C.1 y las demás figuras de este ejemplo abarcarán solamente una de las tres estaciones. No obstante, el análisis se realizará sobre la totalidad de la topología.

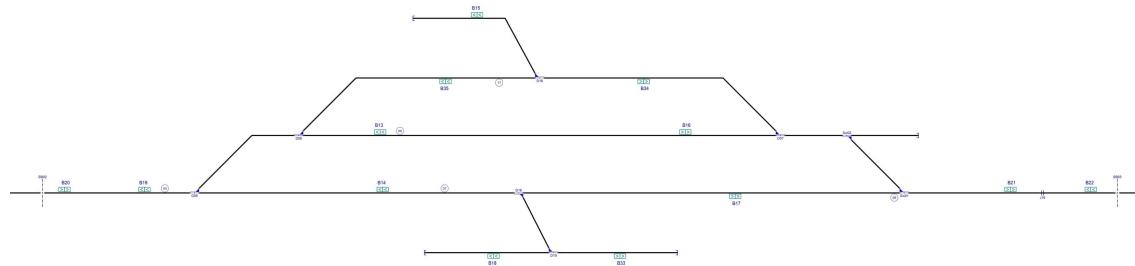


Figura C.1: Topología ferroviaria del ejemplo 4 sin señalamiento.

Esta topología tiene la particularidad de que las tres estaciones son prácticamente idéntica, salvo algunas variaciones. Por ejemplo, la tercera estación incluye un cambio de vías dobles como parte anidada de otros cambios de vías simples, lo cual no se había analizado en otros ejemplos. No obstante, se espera que el señalamiento sea similar en las tres estaciones.

C.2. Señalamiento original

El señalamiento original, ilustrado en la Figura C.2 incluye 71 señales y es demasiado extenso para describirlo en detalle. El mismo incluye señales de parada próximas a los finales de vías absolutos, señales de partida en las plataformas, señales de protección antes de cada paso a nivel señales de maniobras antes de converger en una vía principal y señales múltiples para cambios de vías divergentes.

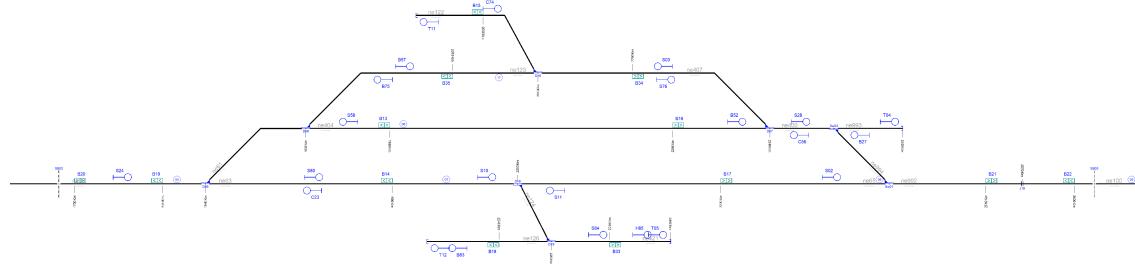


Figura C.2: Señalamiento original del ejemplo 4.

Estas señales permiten definir hasta un máximo de 77 rutas, las cuales no pueden ser detalladas en una única tabla, por lo que serán particionadas en las Tablas C.1, C.2, C.3, C.4 y C.5. En una primera inspección, se puede comprobar que el señalamiento de este ejemplo no es tan complejo como el del ejemplo 3, si es el mas extenso que se analizará en este trabajo.

Tabla C.1: Tabla de enclavamiento original del ejemplo 4 (Rutas 1 a 15).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	S ₁₅	S ₂₄	-	-	Lc ₀₂	ne98-ne99
R ₀₂	S ₁₆	S ₆₇	Sw ₀₄ ^N	-	Lc ₀₂	ne98-ne292
R ₀₃	S ₁₆	S ₄₃	Sw ₀₄ ^R	-	Lc ₀₂	ne98-ne297
R ₀₄	S ₁₉	S ₃₂	-	-	Lc ₀₆	ne100-ne101
R ₀₅	S ₂₀	S ₁₁	Sw ₀₁ ^N	-	Lc ₀₆	ne100-ne65
R ₀₆	S ₂₀	S ₅₆	Sw ₀₁ ^R	-	Lc ₀₆	ne100-ne400
R ₀₇	S ₂₃	S ₁₆	-	-	-	ne63-ne98
R ₀₈	S ₂₄	S ₈₀	D ₀₅ ^N	-	-	ne99-ne63
R ₀₉	S ₂₄	S ₅₂	D ₀₅ ^R -D ₀₈ ^N	-	-	ne99-ne404
R ₁₀	S ₂₄	S ₅₇	D ₀₅ ^R -D ₀₈ ^R	-	-	ne99-ne123
R ₁₁	S ₂₇	S ₅₆	-	-	-	ne993-ne400
R ₁₂	S ₂₈	S ₀₄	Sw ₀₂ ^N	-	-	ne400-ne384
R ₁₃	S ₂₈	S ₁₉	Sw ₀₂ ^R	-	Lc ₀₅	ne400-ne100
R ₁₄	S ₃₁	S ₂₀	-	-	-	ne912-ne100
R ₁₅	S ₃₂	S ₉₁	D ₀₉ ^N	-	-	ne101-ne912

Tabla C.2: Tabla de enclavamiento original del ejemplo 4 (Rutas 16 a 30).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₁₆	S ₃₂	S ₅₉	D ₀₉ ^R -D ₁₀ ^N	-	-	ne ₁₀₁ -ne ₁₃₀
R ₁₇	S ₃₂	S ₆₀	D ₀₉ ^R -D ₁₀ ^R	-	-	ne ₁₀₁ -ne ₁₃₅
R ₁₈	S ₃₅	S ₀₁	-	-	-	ne ₂₉₀ -ne ₁₃₀
R ₁₉	S ₃₉	S ₄₃	-	-	-	ne ₉₉₅ -ne ₂₉₇
R ₂₀	S ₄₄	S ₀₄	-	-	-	ne ₁₁₀ -ne ₃₈₄
R ₂₁	S ₄₅	S ₀₁	-	-	-	ne ₂₉₅ -ne ₁₃₀
R ₂₂	S ₄₆	S ₆₃	Sw ₀₂ ^N	-	-	ne ₃₈₄ -ne ₁₁₀
R ₂₃	S ₄₆	S ₀₉	Sw ₀₂ ^R	-	-	ne ₃₈₄ -ne ₂₉₂
R ₂₄	S ₄₇	S ₄₀	-	-	-	ne ₂₉₅ -ne ₂₉₇
R ₂₅	S ₅₂	S ₂₈	-	-	-	ne ₄₀₄ -ne ₄₀₀
R ₂₆	S ₅₆	S ₅₈	D ₀₇ ^N	-	-	ne ₄₀₀ -ne ₄₀₄
R ₂₇	S ₅₆	S ₀₃	D ₀₇ ^R	-	-	ne ₄₀₀ -ne ₄₀₇
R ₂₈	S ₅₇	S ₇₆	-	-	-	ne ₁₂₃ -ne ₄₀₇
R ₂₉	S ₅₈	S ₁₆	-	-	-	ne ₄₀₄ -ne ₉₈
R ₃₀	S ₅₉	S ₈₇	D ₁₂ ^T	-	-	ne ₁₃₀ -ne ₁₁₄

Tabla C.3: Tabla de enclavamiento original del ejemplo 4 (Rutas 31 a 45).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₃₁	S ₅₉	S ₀₆	D ₁₂ ^S	-	-	ne ₁₃₀ -ne ₃₇₇
R ₃₂	S ₆₀	S ₉₉	-	-	-	ne ₁₃₅ -ne ₁₂₇
R ₃₃	S ₆₂	S ₀₄	-	-	-	ne ₁₀₄ -ne ₃₈₄
R ₃₄	S ₆₃	S ₀₁	-	-	-	ne ₁₁₀ -ne ₁₃₀
R ₃₅	S ₆₇	S ₀₉	-	-	-	ne ₂₉₂ -ne ₂₉₂
R ₃₆	S ₆₈	S ₀₈	-	-	-	ne ₂₉₀ -ne ₂₉₀
R ₃₇	S ₇₄	S ₇₆	-	-	-	ne ₁₂₂ -ne ₄₀₇
R ₃₈	S ₇₅	S ₁₆	-	-	-	ne ₁₂₃ -ne ₉₈
R ₃₉	S ₇₆	S ₂₈	-	-	-	ne ₄₀₇ -ne ₄₀₀
R ₄₀	S ₈₀	S ₁₀	-	-	-	ne ₆₃ -ne ₆₃
R ₄₁	S ₈₃	S ₁₂	-	-	-	ne ₁₂₆ -ne ₁₂₆
R ₄₂	S ₈₄	S ₈₅	-	-	-	ne ₄₂₁ -ne ₄₂₁
R ₄₃	S ₈₅	S ₀₅	-	-	-	ne ₄₂₁ -ne ₄₂₁
R ₄₄	S ₈₆	S ₈₇	-	-	-	ne ₁₃₂ -ne ₁₁₄
R ₄₅	S ₈₇	S ₀₈	-	-	-	ne ₁₁₄ -ne ₂₉₀

Tabla C.4: Tabla de enclavamiento original del ejemplo 4 (Rutas 46 a 160).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₄₆	S ₈₉	S ₃₁	-	-	-	ne ₁₃₂ -ne ₉₁₂
R ₄₇	S ₉₀	S ₈₉	-	-	-	ne ₁₃₂ -ne ₁₃₂
R ₄₈	S ₉₁	S ₉₃	-	-	-	ne ₉₁₂ -ne ₉₁₂
R ₄₉	S ₉₃	S ₈₆	D ₂₀ ^N	-	-	ne ₉₁₂ -ne ₁₃₂
R ₅₀	S ₉₃	S ₉₅	D ₂₀ ^R	-	-	ne ₉₁₂ -ne ₄₆₅
R ₅₁	S ₉₄	S ₁₃	-	-	-	ne ₁₃₃ -ne ₁₃₃
R ₅₂	S ₉₅	S ₉₆	-	-	-	ne ₄₆₅ -ne ₄₆₅
R ₅₃	S ₉₆	S ₀₇	-	-	-	ne ₄₆₅ -ne ₉₉₁
R ₅₄	S ₉₇	S ₉₉	-	-	-	ne ₁₃₄ -ne ₁₂₇
R ₅₅	S ₉₈	S ₂₀	-	-	-	ne ₁₃₅ -ne ₁₀₀
R ₅₆	S ₉₉	S ₈₇	D ₁₂ ^S	-	-	ne ₁₂₇ -ne ₁₁₄
R ₅₇	S ₉₉	S ₀₆	D ₁₂ ^T	-	-	ne ₁₂₇ -ne ₃₇₇
R ₅₈	S ₄₀	S ₀₂	Sw ₀₃ ^N	-	-	ne ₂₉₇ -ne ₆₅
R ₅₉	S ₄₀	S ₁₅	Sw ₀₃ ^R	-	-	ne ₂₉₇ -ne ₉₈
R ₆₀	S ₄₃	S ₄₅	D ₀₄ ^N	-	-	ne ₂₉₇ -ne ₂₉₅

Tabla C.5: Tabla de enclavamiento original del ejemplo 4 (Rutas 61 a 76).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₆₁	S ₄₃	S ₄₆	D ₀₄ ^R	-	-	ne ₂₉₇ -ne ₃₈₄
R ₆₂	S ₀₁	S ₂₀	-	-	-	ne ₁₃₀ -ne ₁₀₀
R ₆₃	S ₀₂	S ₁₉	-	-	Lc ₀₅	ne ₆₅ -ne ₁₀₀
R ₆₄	S ₀₃	S ₇₅	D ₁₆ ^N	-	-	ne ₄₀₇ -ne ₁₂₃
R ₆₅	S ₀₃	S ₁₁	D ₁₆ ^R	-	-	ne ₄₀₇ -ne ₆₅
R ₆₆	S ₀₄	S ₄₀	-	-	-	ne ₃₈₄ -ne ₂₉₇
R ₆₇	S ₀₆	S ₁₀	D ₁₅ ^N	-	-	ne ₃₇₇ -ne ₆₃
R ₆₈	S ₀₆	S ₃₅	D ₁₅ ^R	-	-	ne ₃₇₇ -ne ₂₉₀
R ₆₉	S ₀₇	S ₆₈	D ₀₁ ^N	-	-	ne ₉₉₁ -ne ₂₉₀
R ₇₀	S ₀₇	S ₄₇	D ₀₁ ^R -D ₀₃ ^N	-	-	ne ₉₉₁ -ne ₂₉₅
R ₇₁	S ₀₇	S ₄₄	D ₀₁ ^R -D ₀₃ ^R	-	-	ne ₉₉₁ -ne ₁₁₀
R ₇₂	S ₀₈	S ₁₅	D ₁₄ ^N	-	-	ne ₂₉₀ -ne ₉₈
R ₇₃	S ₀₈	S ₀₃	D ₁₄ ^R	-	-	ne ₂₉₀ -ne ₄₀₇
R ₇₄	S ₀₉	S ₃₅	-	-	-	ne ₂₉₂ -ne ₂₉₀
R ₇₅	S ₁₀	S ₀₂	D ₁₈ ^N	-	-	ne ₆₃ -ne ₆₅
R ₇₆	S ₁₀	S ₈₄	D ₁₈ ^R	-	-	ne ₆₃ -ne ₄₂₁
R ₇₇	S ₁₁	S ₂₃	-	-	-	ne ₆₅ -ne ₆₃

Algunas rutas abarcan mas de un *netElement*, como por ejemplo la ruta R75 que comienza en la señal S10 y finaliza en la señal S02, atravesando los *netElements* ne63 y ne65, utilizando el cambio de vías D18 en posición reversa.

C.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código C.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código C.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analyzing railML object
Analyzing graph
ne991 [-1064, 0] [-2322, 0] <<
ne61 [4350, 0] [4804, 250] >>
ne63 [4350, 0] [5763, 0] >>
ne65 [7444, 0] [5763, 0] <<
ne910 [11254, 0] [11822, 250] >>
ne912 [11254, 0] [13111, 0] >>
ne114 [15670, 0] [14482, 0] <<
```

```

ne288 [-1064, 0] [-741, 250] >>
ne290 [-1064, 0] [-211, 0] >>
ne292 [864, 0] [-211, 0] <<
ne295 [-741, 250] [457, 250] >>
ne297 [457, 250] [720, 250] >>
ne377 [350, -200] [-92, -200] <<
ne384 [457, 250] [-104, 450] <<
ne400 [6904, 250] [7194, 250] >>
ne404 [4804, 250] [6904, 250] >>
ne407 [6904, 250] [5855, 500] <<
ne421 [6454, -259] [5915, -259] <<
ne450 [14722, 250] [14232, 250] <<
ne465 [13772, -259] [13263, -259] <<
ne98 [1573, 0] [3687, 0] >>
ne99 [3687, 0] [4350, 0] >>
ne100 [8372, 0] [10727, 0] >>
ne101 [10727, 0] [11254, 0] >>
ne102 [15670, 0] [18172, 0] >>
ne104 [-509, 630] [-104, 450] >>
ne110 [-104, 450] [-741, 250] <<
ne111 [-211, 0] [-92, -200] >>
ne113 [-92, -200] [-749, -200] <<
ne122 [5304, 759] [5855, 500] >>
ne123 [5855, 500] [4804, 250] <<
ne124 [5763, 0] [5915, -259] >>
ne126 [5915, -259] [5354, -259] <<
ne127 [14232, 250] [13022, 500] <<
ne129 [14232, 250] [14482, 0] >>
ne130 [14232, 250] [11822, 250] <<
ne131 [13111, 0] [13263, -259] >>
ne132 [13111, 0] [14482, 0] >>
ne133 [13263, -259] [12722, -259] <<
ne134 [12452, 810] [13022, 500] >>
ne135 [13022, 500] [11822, 250] <<
ne992 [7444, 0] [8372, 0] >>
ne993 [7194, 250] [7504, 250] >>
ne994 [7444, 0] [7194, 250] <<
ne995 [720, 250] [1020, 250] >>
ne996 [864, 0] [1573, 0] >>
ne997 [720, 250] [864, 0] >>
The network is connected

```

Por ejemplo, el *netElement* ne995 inicia en la coordenada (720;250) y finaliza en la coordenada (1020;250). El símbolo **>>** indica que ne1 se encuentra definido de izquierda a derecha, ya que la componente x de la coordenada final es mayor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura C.2. Por ejemplo, ne61, ne63 y ne99 comparten la coordenada (4350;0), que coincide con la coordenada del cambio de vías D05.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código C.2.

Código C.2: Detección de puntos críticos por parte del RNA

```

Detecting Danger --> Safe_points.RNA
ne991 has a Middle point @ [-2112.3, 0]
ne991 has a Middle point @ [-1902.7, 0]

```

```

ne991 has a Middle point @ [-1693.0, 0]
ne991 has a Middle point @ [-1483.3, 0]
ne991 has a Middle point @ [-1273.7, 0]
ne61 has a Curve(2 lines) @ [[4600, 250]]
ne63 has a Middle point @ [4551.9, 0]
ne63 has a Middle point @ [4753.7, 0]
ne63 has a Middle point @ [4955.6, 0]
ne63 has a Middle point @ [5157.4, 0]
ne63 has a Middle point @ [5359.3, 0]
ne63 has a Middle point @ [5561.1, 0]
ne65 has a Middle point @ [5973.1, 0]
ne65 has a Middle point @ [6183.2, 0]
ne65 has a Middle point @ [6393.4, 0]
ne65 has a Middle point @ [6603.5, 0]
ne65 has a Middle point @ [6813.6, 0]
ne65 has a Middle point @ [7023.8, 0]
ne65 has a Middle point @ [7233.9, 0]
ne910 has a Curve(2 lines) @ [[11504, 250]]
ne912 has a RailJoint[J24] @ [11867, 0]
ne114 has a RailJoint[J27] @ [15294, 0]
ne288 has a Curve(2 lines) @ [[-928, 250]]
ne290 has a RailJoint[J07] @ [-633, 0]
ne292 has a RailJoint[J05] @ [538, 0]
ne295 has a Middle point @ [-501.4, 250]
ne295 has a Middle point @ [-261.8, 250]
ne295 has a Middle point @ [-22.2, 250]
ne295 has a Middle point @ [217.4, 250]
ne297 has a Middle point @ [588.5, 250]
ne377 has a Middle point @ [129.0, -200]
ne384 has a Curve(2 lines) @ [[350, 450]]
ne400 has a Middle point @ [7049.0, 250]
ne404 has a Middle point @ [5014.0, 250]
ne404 has a Middle point @ [5224.0, 250]
ne404 has a Middle point @ [5434.0, 250]
ne404 has a Middle point @ [5644.0, 250]
ne404 has a Middle point @ [5854.0, 250]
ne404 has a Middle point @ [6064.0, 250]
ne404 has a Middle point @ [6274.0, 250]
ne404 has a Middle point @ [6484.0, 250]
ne404 has a Middle point @ [6694.0, 250]
ne407 has a Curve(2 lines) @ [[6654, 500]]
ne421 has a Middle point @ [6184.5, -259]
ne450 has a Middle point @ [14477.0, 250]
ne465 has a Middle point @ [13517.5, -259]
ne98 has a LevelCrossing[lcr495] @ [2419, 0]
ne98 has a LevelCrossing[lcr496] @ [2419, 0]
ne99 has a Middle point @ [3908.0, 0]
ne99 has a Middle point @ [4129.0, 0]
ne100 has a LevelCrossing[lcr497] @ [9346, 0]
ne100 has a LevelCrossing[lcr498] @ [10127, 0]
ne100 has a LevelCrossing[lcr499] @ [9347, 0]
ne100 has a LevelCrossing[lcr500] @ [10127, 0]
ne101 has a RailJoint[J26] @ [10914, 0]
ne102 has a Middle point @ [15878.5, 0]
ne102 has a Middle point @ [16087.0, 0]
ne102 has a Middle point @ [16295.5, 0]
ne102 has a Middle point @ [16504.0, 0]
ne102 has a Middle point @ [16712.5, 0]
ne102 has a Middle point @ [16921.0, 0]
ne102 has a Middle point @ [17129.5, 0]
ne102 has a Middle point @ [17338.0, 0]
ne102 has a Middle point @ [17546.5, 0]
ne102 has a Middle point @ [17755.0, 0]
ne102 has a Middle point @ [17963.5, 0]

```

```

ne104 has a Curve(2 lines) @ [[-209, 630]]
ne110 has a Curve(2 lines) @ [[-633, 450]]
ne113 has a Middle point @ [-530.0, -200]
ne113 has a Middle point @ [-311.0, -200]
ne122 has a Curve(2 lines) @ [[5704, 759]]
ne123 has a Curve(2 lines) @ [[5054, 500]]
ne126 has a Middle point @ [5634.5, -259]
ne127 has a Curve(2 lines) @ [[13982, 500]]
ne130 has a Middle point @ [12022.8, 250]
ne130 has a Middle point @ [12223.7, 250]
ne130 has a Middle point @ [12424.5, 250]
ne130 has a Middle point @ [12625.3, 250]
ne130 has a Middle point @ [12826.2, 250]
ne130 has a Middle point @ [13027.0, 250]
ne130 has a Middle point @ [13227.8, 250]
ne130 has a Middle point @ [13428.7, 250]
ne130 has a Middle point @ [13629.5, 250]
ne130 has a Middle point @ [13830.3, 250]
ne130 has a Middle point @ [14031.2, 250]
ne132 has a RailJoint[J21] @ [14309, 0]
ne133 has a Middle point @ [12992.5, -259]
ne134 has a Curve(2 lines) @ [[12842, 810]]
ne135 has a Curve(2 lines) @ [[12072, 500]]
ne992 has a RailJoint[J19] @ [8046, 0]
ne993 has a Middle point @ [7349.0, 250]
ne995 has a Middle point @ [870.0, 250]
ne996 has a RailJoint[J08] @ [1094, 0]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código C.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código C.3: Infrastructure.RNA

```

Nodes: 47|Switches: 23|Signals: 0|Detectors: 8|Ends: 19|Barriers: 2
Node ne91:
    Track = track1
    Type = BufferStop -> ['E16']
    Neighbours = 2 -> ['ne288', 'ne290']
    Switches -> D01
        ContinueCourse -> right -> ne290
        BranchCourse -> left -> ne288
Node ne61:
    Track = track17
    Neighbours = 4 -> ['ne99', 'ne63', 'ne404', 'ne123']
    Switches -> D08
        ContinueCourse -> right -> ne404
        BranchCourse -> left -> ne123
Node ne63:
    Track = track16
    Neighbours = 4 -> ['ne61', 'ne99', 'ne65', 'ne124']
    Switches -> D18
        ContinueCourse -> left -> ne65
        BranchCourse -> right -> ne124
Node ne65:
    Track = track35
    Neighbours = 4 -> ['ne63', 'ne992', 'ne994', 'ne124']
Node ne910:
    Track = track20
    Neighbours = 4 -> ['ne912', 'ne101', 'ne130', 'ne135']
    Switches -> D10
        ContinueCourse -> right -> ne130
        BranchCourse -> left -> ne135
Node ne912:

```

```

Track = track19
TrainDetectionElements -> tde527
    Type -> insulatedRailJoint
    Side -> left
Neighbours = 4 -> ['ne910', 'ne101', 'ne131', 'ne132']
Switches -> D20
    ContinueCourse -> left -> ne132
    BranchCourse -> right -> ne131
Node ne114:
    TrainDetectionElements -> tde534
    Type -> insulatedRailJoint
    Side -> left
    Neighbours = 3 -> ['ne132', 'ne129', 'ne102']
    Switches -> D23
        ContinueCourse -> left -> ne132
        BranchCourse -> right -> ne129
Node ne288:
    Track = track22
    Neighbours = 4 -> ['ne991', 'ne290', 'ne295', 'ne110']
    Switches -> D03
        ContinueCourse -> right -> ne295
        BranchCourse -> left -> ne110
Node ne290:
    Track = track21
    TrainDetectionElements -> tde502
        Type -> insulatedRailJoint
        Side -> left
    Neighbours = 4 -> ['ne991', 'ne288', 'ne292', 'ne111']
    Switches -> D14
        ContinueCourse -> left -> ne292
        BranchCourse -> right -> ne111
Node ne292:
    Track = track33
    TrainDetectionElements -> tde511
        Type -> insulatedRailJoint
        Side -> right
    Neighbours = 4 -> ['ne290', 'ne996', 'ne997', 'ne111']
Node ne295:
    Track = track23
    Neighbours = 4 -> ['ne288', 'ne297', 'ne384', 'ne110']
Node ne297:
    Track = track25
    Neighbours = 4 -> ['ne295', 'ne384', 'ne995', 'ne997']
    Switches -> D04
        ContinueCourse -> left -> ne295
        BranchCourse -> right -> ne384
    Switches -> Sw03
        ContinueCourse -> left -> ne995
        BranchCourse -> right -> ne997
Node ne377:
    Track = track3
    Type = BufferStop -> ['E305']
    Neighbours = 2 -> ['ne111', 'ne113']
    Switches -> D15
        ContinueCourse -> left -> ne113
        BranchCourse -> right -> ne111
Node ne384:
    Track = track26
    Neighbours = 4 -> ['ne295', 'ne297', 'ne104', 'ne110']
Node ne400:
    Track = track27
    Neighbours = 4 -> ['ne404', 'ne407', 'ne993', 'ne994']
    Switches -> D07
        ContinueCourse -> left -> ne404

```

```

        BranchCourse -> right -> ne407
        Switches -> Sw02
        ContinueCourse -> left -> ne993
        BranchCourse -> right -> ne994
Node ne404:
    Track = track28
    Neighbours = 4 -> ['ne61', 'ne400', 'ne407', 'ne123']
Node ne407:
    Track = track29
    Neighbours = 4 -> ['ne400', 'ne404', 'ne122', 'ne123']
    Switches -> D16
        ContinueCourse -> left -> ne123
        BranchCourse -> right -> ne122
Node ne421:
    Track = track8
    Type = BufferStop -> ['E418']
    Neighbours = 2 -> ['ne124', 'ne126']
    Switches -> D19
        ContinueCourse -> left -> ne126
        BranchCourse -> right -> ne124
Node ne450:
    Track = track12
    Type = BufferStop -> ['E478']
    Neighbours = 3 -> ['ne127', 'ne129', 'ne130']
Node ne465:
    Track = track10
    Type = BufferStop -> ['E462']
    Neighbours = 2 -> ['ne131', 'ne133']
    Switches -> D21
        ContinueCourse -> left -> ne133
        BranchCourse -> right -> ne131
Node ne98:
    Neighbours = 2 -> ['ne996', 'ne99']
    Level crossing -> Lc01
        Protection -> true | Barriers -> none | Lights -> none Acoustic -> none
        Position -> [2464, 0] | Coordinate: 0.4218
Node ne99:
    Track = track15
    Neighbours = 3 -> ['ne61', 'ne63', 'ne98']
    Switches -> D05
        ContinueCourse -> right -> ne63
        BranchCourse -> left -> ne61
Node ne100:
    Neighbours = 2 -> ['ne992', 'ne101']
    Level crossing -> Lc04
        Protection -> true | Barriers -> none | Lights -> none Acoustic -> none
        Position -> [10172, 0] | Coordinate: 0.7643
Node ne101:
    Track = track18
    TrainDetectionElements -> tde526
        Type -> insulatedRailJoint
        Side -> left
    Neighbours = 3 -> ['ne910', 'ne912', 'ne100']
    Switches -> D09
        ContinueCourse -> right -> ne912
        BranchCourse -> left -> ne910
Node ne102:
    Track = track2
    Type = BufferStop -> ['E109']
    Neighbours = 1 -> ['ne114']
Node ne104:
    Track = track13
    Type = BufferStop -> ['bus541']
    Neighbours = 2 -> ['ne384', 'ne110']

```

```

Node ne110:
    Track = track24
    Neighbours = 4 -> ['ne288', 'ne295', 'ne384', 'ne104']
Node ne111:
    Track = track34
    Neighbours = 4 -> ['ne290', 'ne292', 'ne377', 'ne113']
Node ne113:
    Track = track5
    Type = BufferStop -> ['E368']
    Neighbours = 2 -> ['ne377', 'ne111']
Node ne122:
    Track = track6
    Type = BufferStop -> ['E408']
    Neighbours = 2 -> ['ne407', 'ne123']
Node ne123:
    Track = track30
    Neighbours = 4 -> ['ne61', 'ne404', 'ne407', 'ne122']
Node ne124:
    Track = track36
    Neighbours = 4 -> ['ne63', 'ne65', 'ne421', 'ne126']
Node ne126:
    Track = track9
    Type = BufferStop -> ['E422']
    Neighbours = 2 -> ['ne421', 'ne124']
Node ne127:
    Track = track39
    Neighbours = 5 -> ['ne450', 'ne129', 'ne130', 'ne135', 'ne134']
    Switches -> D12A
        ContinueCourse -> right -> ne450
        BranchCourse -> left -> ne129
    Switches -> D24
        ContinueCourse -> left -> ne135
        BranchCourse -> right -> ne134
Node ne129:
    Track = track38
    Neighbours = 5 -> ['ne114', 'ne450', 'ne127', 'ne130', 'ne132']
Node ne130:
    Track = track31
    Neighbours = 5 -> ['ne910', 'ne450', 'ne127', 'ne129', 'ne135']
    Switches -> D12B
        ContinueCourse -> right -> ne129
        BranchCourse -> left -> ne450
Node ne131:
    Track = track40
    Neighbours = 4 -> ['ne912', 'ne465', 'ne132', 'ne133']
Node ne132:
    Track = track37
    TrainDetectionElements -> tde533
        Type -> insulatedRailJoint
        Side -> right
    Neighbours = 4 -> ['ne912', 'ne114', 'ne129', 'ne131']
Node ne133:
    Track = track11
    Type = BufferStop -> ['E466']
    Neighbours = 2 -> ['ne465', 'ne131']
Node ne134:
    Track = track14
    Type = BufferStop -> ['bus570']
    Neighbours = 2 -> ['ne127', 'ne135']
Node ne135:
    Track = track32
    Neighbours = 4 -> ['ne910', 'ne127', 'ne130', 'ne134']
Node ne992:
    TrainDetectionElements -> tde525

```

```

Type -> insulatedRailJoint
Side -> right
Neighbours = 3 -> ['ne65', 'ne100', 'ne994']
Switches -> Sw01
ContinueCourse -> left -> ne65
BranchCourse -> right -> ne994
Node ne993:
    Track = track7
    Type = BufferStop -> ['E412']
    Neighbours = 2 -> ['ne400', 'ne994']
Node ne994:
    Track = track41
    Neighbours = 4 -> ['ne65', 'ne400', 'ne992', 'ne993']
Node ne995:
    Track = track4
    Type = BufferStop -> ['E313']
    Neighbours = 2 -> ['ne297', 'ne997']
Node ne996:
    TrainDetectionElements -> tde514
    Type -> insulatedRailJoint
    Side -> right
    Neighbours = 3 -> ['ne292', 'ne98', 'ne997']
    Switches -> Sw04
        ContinueCourse -> left -> ne292
        BranchCourse -> right -> ne997
Node ne997:
    Track = track42
    Neighbours = 4 -> ['ne292', 'ne297', 'ne995', 'ne996']

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código C.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código C.4: SafePoints.RNA

```

ne991:
    Next: [[-2112.3, 0], [-1902.7, 0], [-1693.0, 0], [-1483.3, 0], [-1273.7, 0]]
    Prev: [[-2112.3, 0], [-1902.7, 0], [-1693.0, 0], [-1483.3, 0], [-1273.7, 0]]
ne61:
    Prev: [[4700.0, 250]]
ne63:
    Next: [[4551.9, 0], [4753.7, 0], [4955.6, 0], [5157.4, 0], [5359.3, 0], [5561.1, 0]]
    Prev: [[4551.9, 0], [4753.7, 0], [4955.6, 0], [5157.4, 0], [5359.3, 0], [5561.1, 0]]
ne65:
    Next: [[5973.1, 0], [6183.2, 0], [6393.4, 0], [6603.5, 0], [6813.6, 0], [7023.8, 0], [7233.9,
        ↪ 0]]
    Prev: [[5973.1, 0], [6183.2, 0], [6393.4, 0], [6603.5, 0], [6813.6, 0], [7023.8, 0], [7233.9,
        ↪ 0]]
ne910:
    Prev: [[11604.0, 250]]
ne912:
    Next: [[11767.0, 0]]
    Prev: [[11967.0, 0]]
ne114:
    Next: [[15194.0, 0]]
    Prev: [[15394.0, 0]]
ne288:

```

```

    Prev: [[-828.0, 250]]
ne290:
    Next: [[-733.0, 0]]
    Prev: [[-533.0, 0]]
ne292:
    Next: [[438.0, 0]]
    Prev: [[638.0, 0]]
ne295:
    Next: [[-501.4, 250], [-261.8, 250], [-22.2, 250], [217.4, 250]]
    Prev: [[-501.4, 250], [-261.8, 250], [-22.2, 250], [217.4, 250]]
ne297:
    Next: [[588.5, 250]]
    Prev: [[588.5, 250]]
ne377:
    Next: [[129.0, -200]]
    Prev: [[129.0, -200]]
ne384:
    Next: [[250.0, 450]]
ne400:
    Next: [[7049.0, 250]]
    Prev: [[7049.0, 250]]
ne404:
    Next: [[5014.0, 250], [5224.0, 250], [5434.0, 250], [5644.0, 250], [5854.0, 250], [6064.0, 250],
           ↪ [6274.0, 250], [6484.0, 250], [6694.0, 250]]
    Prev: [[5014.0, 250], [5224.0, 250], [5434.0, 250], [5644.0, 250], [5854.0, 250], [6064.0, 250],
           ↪ [6274.0, 250], [6484.0, 250], [6694.0, 250]]
ne407:
    Next: [[6554.0, 500]]
ne421:
    Next: [[6184.5, -259]]
    Prev: [[6184.5, -259]]
ne450:
    Next: [[14477.0, 250]]
    Prev: [[14477.0, 250]]
ne465:
    Next: [[13817.5, -259]]
    Prev: [[13517.5, -259]]
ne98:
    Next: [[2219, 0]]
    Prev: [[2619, 0]]
ne99:
    Next: [[3908.0, 0], [4129.0, 0]]
    Prev: [[3908.0, 0], [4129.0, 0]]
ne100:
    Next: [[9927, 0]]
    Prev: [[10327, 0]]
ne101:
    Next: [[10814.0, 0]]
    Prev: [[11014.0, 0]]
ne102:
    Next: [[15878.5, 0], [16087.0, 0], [16295.5, 0], [16504.0, 0], [16712.5, 0], [16921.0, 0],
           ↪ [17129.5, 0], [17338.0, 0], [17546.5, 0], [17755.0, 0], [17963.5, 0]]
    Prev: [[15878.5, 0], [16087.0, 0], [16295.5, 0], [16504.0, 0], [16712.5, 0], [16921.0, 0],
           ↪ [17129.5, 0], [17338.0, 0], [17546.5, 0], [17755.0, 0], [17963.5, 0]]
ne104:
    Next: [[-309.0, 630]]
ne110:
    Prev: [[-533.0, 450]]
ne113:
    Next: [[-530.0, -200], [-311.0, -200]]
    Prev: [[-530.0, -200], [-311.0, -200]]
ne122:
    Next: [[5604.0, 759]]
ne123:

```

```

Prev: [[5154.0, 500]]
ne126:
  Next: [[5634.5, -259]]
  Prev: [[5634.5, -259]]
ne127:
  Next: [[13882.0, 500]]
ne130:
  Next: [[12022.8, 250], [12223.7, 250], [12424.5, 250], [12625.3, 250], [12826.2, 250], [13027.0,
    ↪ 250], [13227.8, 250], [13428.7, 250], [13629.5, 250], [13830.3, 250], [14031.2, 250]]
  Prev: [[12022.8, 250], [12223.7, 250], [12424.5, 250], [12625.3, 250], [12826.2, 250], [13027.0,
    ↪ 250], [13227.8, 250], [13428.7, 250], [13629.5, 250], [13830.3, 250], [14031.2, 250]]
ne132:
  Next: [[14209.0, 0]]
  Prev: [[14409.0, 0]]
ne133:
  Next: [[12992.5, -259]]
  Prev: [[12992.5, -259]]
ne134:
  Next: [[12742.0, 810]]
ne135:
  Prev: [[12172.0, 500]]
ne992:
  Next: [[7946.0, 0]]
  Prev: [[8146.0, 0]]
ne993:
  Next: [[7349.0, 250]]
  Prev: [[7349.0, 250]]
ne995:
  Next: [[870.0, 250]]
  Prev: [[870.0, 250]]
ne996:
  Next: [[994.0, 0]]
  Prev: [[1194.0, 0]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura C.3.

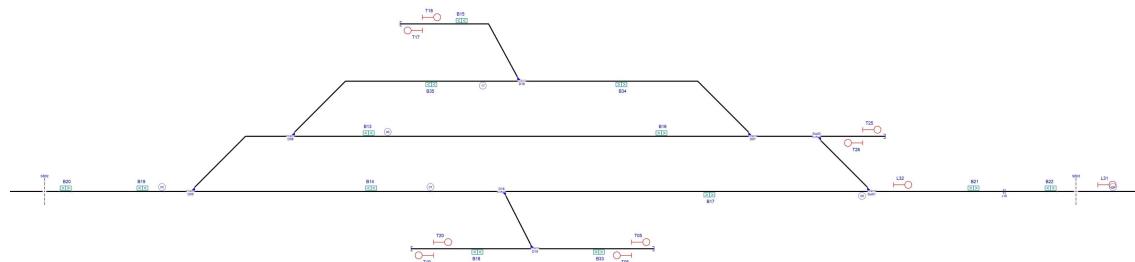


Figura C.3: Señalamiento generado por el RNA para proteger el fin de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01, T03, T05, T07, T09, T11, T13, T15, T17, T19, T21, T23, T25 y T27; y las señales de partida son T02, T04, T06, T08, T10, T12, T14, T16, T18, T20, T22, T24, T26 y T28. A su vez, al no existir finales de vías relativos, el RNA no les asignó señales específicas.

La Figura C.4 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Estas señales se obtuvieron al aplicar el Algoritmo 8, tal como fue explicado en la Sección 2.2.2. Las señales generadas son todas las señales entre J34 y J49, indicadas en color rojo. De no existir junturas que proteger, el RNA saltará este paso.

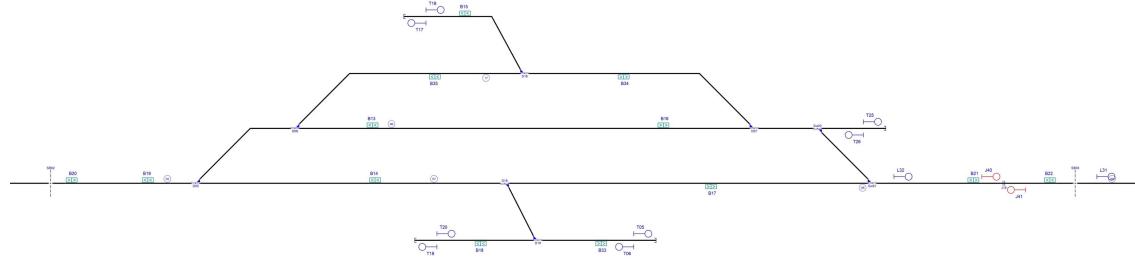


Figura C.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos. El señalamiento generado para proteger las plataformas y los cruces de vía, producto de aplicar el Algoritmo 9 y el Algoritmo 10, respectivamente, se ilustra en rojo en la Figura 4.5. No se asignaron señales para proteger las plataformas, al no existir en este ejemplo. Por otro lado, las señales que protegen los cruces de vía son las señales comprendidas entre X50 y X53.

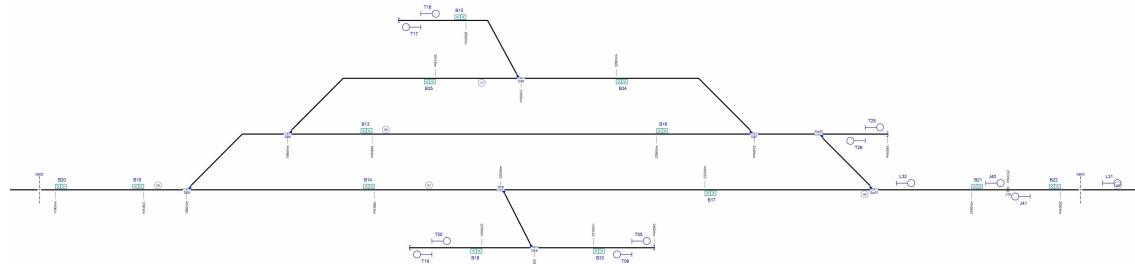


Figura C.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA generó 75 señales para proteger cada uno de los cambio de vías. Estas señales se encuentran resaltadas en rojo en la Figura C.6.

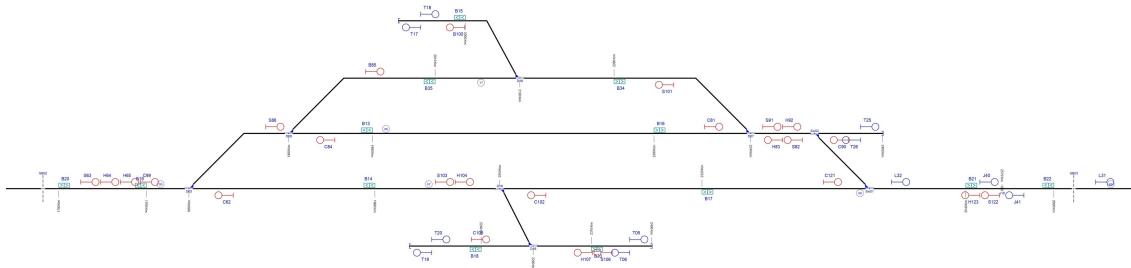


Figura C.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías que comparan al menos una rama secundaria, desplazando las señales hasta convertirlas en las señales de herencia en las ramas principales. A continuación, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código C.5.

Código C.5: Reducción de señalamiento por prioridad de señales

```

Reducing redundant signals
removing sig97 for sig04
removing sig98 for sig04
removing sig106 for sig06
removing sig107 for sig06
removing sig115 for sig10
removing sig116 for sig10
removing sig96 for sig16
removing sig100 for sig17
removing sig105 for sig20
removing sig114 for sig22
removing sig118 for sig23
removing sig90 for sig26
removing sig124 for sig28
removing sig32 for sig40
removing sig33 for sig38
removing sig34 for sig94
removing sig70 for sig35
removing sig127 for sig36
removing sig93 for sig37
removing sig39 for sig128
removing sig41 for sig122
removing sig42 for sig67
removing sig44 for sig112
removing sig66 for sig45
removing sig108 for sig46
removing sig111 for sig47
removing sig49 for sig109
removing sig50 for sig52
removing sig51 for sig53
removing sig54 for sig58
removing sig55 for sig59
removing sig56 for sig60
removing sig57 for sig61

```

```

removing sig99 for sig63
removing sig64 for sig63
removing sig65 for sig63
removing sig117 for sig67
removing sig68 for sig67
removing sig69 for sig67
removing sig72 for sig71
removing sig73 for sig71
removing sig80 for sig79
removing sig83 for sig82
removing sig92 for sig91
removing sig95 for sig94
removing sig104 for sig103
removing sig110 for sig109
removing sig113 for sig112
removing sig120 for sig119
removing sig123 for sig122
removing sig126 for sig125
removing sig129 for sig128

```

El resultado de la simplificación del señalamiento se ilustra en la Figura C.7.

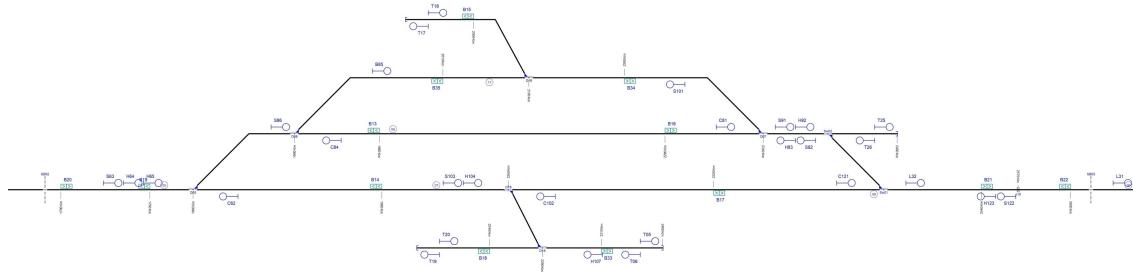


Figura C.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código C.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código C.6: Signalling.RNA

```

T01 [T01] <<:
From: ne991 | To: E16_left
Type: Stop | Direction: normal | AtTrack: left
Position: [-2222, 0] | Coordinate: 0.0794
T02 [T02] >>:
From: ne991 | To: ne991_right
Type: Stop | Direction: reverse | AtTrack: right
Position: [-2222, 0] | Coordinate: 0.0794
T03 [T03] >>:
From: ne377 | To: E305_right
Type: Stop | Direction: reverse | AtTrack: right
Position: [250, 200] | Coordinate: 0.7737
T04 [T04] <<:
From: ne377 | To: ne377_left
Type: Stop | Direction: normal | AtTrack: left
Position: [250, 200] | Coordinate: 0.7737
T05 [T05] >>:
From: ne421 | To: E418_right
Type: Stop | Direction: reverse | AtTrack: right

```

```

Position: [6354, 259] | Coordinate: 0.8144
T06 [T06] <<:
  From: ne421 | To: ne421_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [6354, 259] | Coordinate: 0.8144
T07 [T07] >>:
  From: ne450 | To: E478_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [14622, -250] | Coordinate: 0.7959
T08 [T08] <<:
  From: ne450 | To: ne450_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [14622, -250] | Coordinate: 0.7959
T09 [T09] >>:
  From: ne465 | To: E462_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [13672, 259] | Coordinate: 0.8035
T10 [T10] <<:
  From: ne465 | To: ne465_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [13672, 259] | Coordinate: 0.8035
T11 [T11] >>:
  From: ne102 | To: E109_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [18072, 0] | Coordinate: 0.9600
T12 [T12] <<:
  From: ne102 | To: ne102_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [18072, 0] | Coordinate: 0.9600
T13 [T13] <<:
  From: ne104 | To: bus541_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-409, -630] | Coordinate: 0.6065
T14 [T14] >>:
  From: ne104 | To: ne104_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-409, -630] | Coordinate: 0.6065
T15 [T15] <<:
  From: ne113 | To: E368_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-649, 200] | Coordinate: 0.1522
T16 [T16] >>:
  From: ne113 | To: ne113_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-649, 200] | Coordinate: 0.1522
T17 [T17] <<:
  From: ne122 | To: E408_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [5404, -759] | Coordinate: 0.5713
T18 [T18] >>:
  From: ne122 | To: ne122_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [5404, -759] | Coordinate: 0.5713
T19 [T19] <<:
  From: ne126 | To: E422_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [5454, 259] | Coordinate: 0.1782
T20 [T20] >>:
  From: ne126 | To: ne126_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [5454, 259] | Coordinate: 0.1782
T21 [T21] <<:
  From: ne133 | To: E466_left
  Type: Stop | Direction: normal | AtTrack: left

```

```

Position: [12822, 259] | Coordinate: 0.1848
T22 [T22] >>:
From: ne133 | To: ne133_right
Type: Stop | Direction: reverse | AtTrack: right
Position: [12822, 259] | Coordinate: 0.1848
T23 [T23] <<:
From: ne134 | To: bus570_left
Type: Stop | Direction: reverse | AtTrack: right
Position: [12552, -810] | Coordinate: 0.6125
T24 [T24] >>:
From: ne134 | To: ne134_right
Type: Stop | Direction: normal | AtTrack: left
Position: [12552, -810] | Coordinate: 0.6125
T25 [T25] >>:
From: ne993 | To: E412_right
Type: Stop | Direction: normal | AtTrack: left
Position: [7404, -250] | Coordinate: 0.6774
T27 [T27] >>:
From: ne995 | To: E313_right
Type: Stop | Direction: normal | AtTrack: left
Position: [920, -250] | Coordinate: 0.6666
L29 [L29] >>:
From: ne114 | To: sb540_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [14582, 0] | Coordinate: 0.0841
L30 [L30] >>:
From: ne98 | To: sb537_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [1673, 0] | Coordinate: 0.0473
L31 [L31] >>:
From: ne100 | To: sb539_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [8472, 0] | Coordinate: 0.0424
J35 [J35] <<:
From: ne290 | To: ne290_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-533.0, 0] | Coordinate: 0.6225
J36 [J36] >>:
From: ne292 | To: ne292_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [438.0, 0] | Coordinate: 0.6037
J37 [J37] <<:
From: ne292 | To: ne292_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [638.0, 0] | Coordinate: 0.7897
J38 [J38] >>:
From: ne996 | To: ne996_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [994.0, 0] | Coordinate: 0.1833
J40 [J40] >>:
From: ne992 | To: ne992_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [7946.0, 0] | Coordinate: 0.5409
J43 [J43] <<:
From: ne101 | To: ne101_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [11014.0, 0] | Coordinate: 0.5445
J45 [J45] <<:
From: ne912 | To: ne912_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [11967.0, 0] | Coordinate: 0.3839
J46 [J46] >>:
From: ne132 | To: ne132_right
Type: Circulation | Direction: normal | AtTrack: left

```

```

Position: [14209.0, 0] | Coordinate: 0.8008
J47 [J47] <<:
From: ne132 | To: ne132_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [14409.0, 0] | Coordinate: 0.9467
J48 [J48] >>:
From: ne114 | To: ne114_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [15194.0, 0] | Coordinate: 0.5993
X50 [X50] >>:
From: ne98 | To: ne98_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [2219, 0] | Coordinate: 0.3055
X51 [X51] <<:
From: ne98 | To: ne98_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [2619, 0] | Coordinate: 0.4947
X52 [X52] >>:
From: ne100 | To: ne100_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [9927, 0] | Coordinate: 0.6602
X53 [X53] <<:
From: ne100 | To: ne100_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [10327, 0] | Coordinate: 0.8301
C54 [C54] <<:
From: ne63 | To: ne63_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [4551.9, 0] | Coordinate: 0.1428
S55 [S55] >>:
From: ne99 | To: ne99_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [4129.0, 0] | Coordinate: 0.6666
S59 [S59] >>:
From: ne101 | To: ne101_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [11014.0, 0] | Coordinate: 0.5445
S64 [S64] >>:
From: ne991 | To: ne991_right
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-1273.7, 0] | Coordinate: 0.8333
C67 [C67] <<:
From: ne295 | To: ne295_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [-501.4, -250] | Coordinate: 0.2
B68 [B68] >>:
From: ne110 | To: ne110_left
Type: Manouver | Direction: reverse | AtTrack: right
Position: [-533.0, -450] | Coordinate: 0.4327
S69 [S69] >>:
From: ne288 | To: ne288_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [-828.0, -250] | Coordinate: 0.8155
C70 [C70] >>:
From: ne295 | To: ne295_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [217.4, -250] | Coordinate: 0.7999
B71 [B71] <<:
From: ne384 | To: ne384_right
Type: Manouver | Direction: normal | AtTrack: left
Position: [250.0, -450] | Coordinate: 0.8531
S72 [S72] <<:
From: ne297 | To: ne297_left
Type: Circulation | Direction: reverse | AtTrack: right

```

```

Position: [588.5, -250] | Coordinate: 0.5
C74 [C74] >>:
From: ne404 | To: ne404_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [6694.0, -250] | Coordinate: 0.9
S75 [S75] <<:
From: ne400 | To: ne400_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [7049.0, -250] | Coordinate: 0.5
C77 [C77] <<:
From: ne404 | To: ne404_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [5014.0, -250] | Coordinate: 0.1
B78 [B78] >>:
From: ne123 | To: ne123_left
Type: Manouver | Direction: reverse | AtTrack: right
Position: [5154.0, -500] | Coordinate: 0.3928
S79 [S79] >>:
From: ne61 | To: ne61_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [4700.0, -250] | Coordinate: 0.8134
C80 [C80] <<:
From: ne130 | To: ne130_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [12022.8, -250] | Coordinate: 0.0833
B81 [B81] >>:
From: ne135 | To: ne135_left
Type: Manouver | Direction: reverse | AtTrack: right
Position: [12172.0, -500] | Coordinate: 0.3479
S82 [S82] >>:
From: ne910 | To: ne910_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [11604.0, -250] | Coordinate: 0.6753
C83 [C83] <<:
From: ne993 | To: ne993_left
Type: Circulation | Direction: reverse | AtTrack: right
Position: [7349.0, -250] | Coordinate: 0.5
S84 [S84] >>:
From: ne400 | To: ne400_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [7049.0, -250] | Coordinate: 0.5
S87 [S87] >>:
From: ne290 | To: ne290_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [-533.0, 0] | Coordinate: 0.6225
S94 [S94] <<:
From: ne407 | To: ne407_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [6554.0, -500] | Coordinate: 0.9132
C95 [C95] <<:
From: ne65 | To: ne65_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [5973.1, 0] | Coordinate: 0.1249
S96 [S96] >>:
From: ne63 | To: ne63_right
Type: Circulation | Direction: normal | AtTrack: left
Position: [5561.1, 0] | Coordinate: 0.8571
S102 [S102] <<:
From: ne114 | To: ne114_left
Type: Circulation | Direction: normal | AtTrack: left
Position: [15194.0, 0] | Coordinate: 0.5993
S105 [S105] >>:
From: ne127 | To: ne127_right
Type: Circulation | Direction: reverse | AtTrack: right

```

```

Position: [13882.0, -500] | Coordinate: 0.9238
S108 [S108] >>:
  From: ne130 | To: ne130_right
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [14031.2, -250] | Coordinate: 0.9166
S111 [S111] >>:
  From: ne912 | To: ne912_right
  Type: Circulation | Direction: normal | AtTrack: left
  Position: [11967.0, 0] | Coordinate: 0.3839
S118 [S118] <<:
  From: ne127 | To: ne127_left
  Type: Circulation | Direction: normal | AtTrack: left
  Position: [13882.0, -500] | Coordinate: 0.9238
C120 [C120] >>:
  From: ne65 | To: ne65_right
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [7233.9, 0] | Coordinate: 0.8750
S121 [S121] <<:
  From: ne992 | To: ne992_left
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [7946.0, 0] | Coordinate: 0.5409
C123 [C123] <<:
  From: ne995 | To: ne995_left
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [870.0, -250] | Coordinate: 0.5
S124 [S124] >>:
  From: ne297 | To: ne297_right
  Type: Circulation | Direction: normal | AtTrack: left
  Position: [588.5, -250] | Coordinate: 0.5
S127 [S127] <<:
  From: ne996 | To: ne996_left
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [994.0, 0] | Coordinate: 0.1833

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código C.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código C.7: Routes.RNA

```

route_1 [T02 >> S64]:
  Path: ['ne991']
route_2 [T04 << J35]:
  Path: ['ne377', 'ne111', 'ne290']
  Switches: ['D14_R', 'D15_R']
route_3 [T04 << T15]:
  Path: ['ne377', 'ne113']
  Switches: ['D15_N']
route_4 [T06 << C54]:
  Path: ['ne421', 'ne124', 'ne63']
  Switches: ['D18_R', 'D19_R']
route_5 [T06 << T19]:
  Path: ['ne421', 'ne126']
  Switches: ['D19_N']
route_6 [T08 << S118]:
  Path: ['ne450', 'ne127']
  Switches: ['D12_RN']
route_7 [T08 << C80]:
  Path: ['ne450', 'ne130']
  Switches: ['D12_NN']

```

```

route_8 [T10 << J45]:
    Path: ['ne465', 'ne131', 'ne912']
    Switches: ['D20_R', 'D21_R']
route_9 [T10 << T21]:
    Path: ['ne465', 'ne133']
    Switches: ['D21_N']
route_10 [T12 << S102]:
    Path: ['ne102', 'ne114']
route_11 [T14 >> S124]:
    Path: ['ne104', 'ne384', 'ne297']
    Switches: ['D04_R']
route_12 [T16 >> T03]:
    Path: ['ne113', 'ne377']
    Switches: ['D15_N']
route_13 [T18 >> S84]:
    Path: ['ne122', 'ne407', 'ne400']
    Switches: ['D07_R', 'D16_R']
route_14 [T20 >> T05]:
    Path: ['ne126', 'ne421']
    Switches: ['D19_N']
route_15 [T22 >> T09]:
    Path: ['ne133', 'ne465']
    Switches: ['D21_N']
route_16 [T24 >> S105]:
    Path: ['ne134', 'ne127']
    Switches: ['D24_R']
route_17 [L29 >> J48]:
    Path: ['ne114']
route_18 [L30 >> X50]:
    Path: ['ne98']
route_19 [L31 >> X52]:
    Path: ['ne100']
route_20 [J35 << T01]:
    Path: ['ne290', 'ne991']
    Switches: ['D01_N']
route_21 [J36 >> J38]:
    Path: ['ne292', 'ne996']
    Switches: ['Sw04_N']
route_22 [J37 << J35]:
    Path: ['ne292', 'ne290']
    Switches: ['D14_N']
route_23 [J38 >> L30]:
    Path: ['ne996', 'ne98']
route_24 [J40 >> L31]:
    Path: ['ne992', 'ne100']
route_25 [J43 << X53]:
    Path: ['ne101', 'ne100']
route_26 [J45 << J43]:
    Path: ['ne912', 'ne101']
    Switches: ['D09_N']
route_27 [J46 >> L29]:
    Path: ['ne132', 'ne114']
    Switches: ['D23_N']
route_28 [J47 << J45]:
    Path: ['ne132', 'ne912']
    Switches: ['D20_N']
route_29 [J48 >> T11]:
    Path: ['ne114', 'ne102']
route_30 [X50 >> S55]:
    Path: ['ne98', 'ne99']
    LevelCrossings: ['Lc01']
route_31 [X51 << S127]:
    Path: ['ne98', 'ne996']
    LevelCrossings: ['Lc01']

```

```

route_32 [X52 >> S59]:
    Path: ['ne100', 'ne101']
    LevelCrossings: ['Lc04']
route_33 [X53 << S121]:
    Path: ['ne100', 'ne992']
    LevelCrossings: ['Lc04']
route_34 [C54 << X51]:
    Path: ['ne63', 'ne99', 'ne98']
    Switches: ['D05_N']
route_35 [S55 >> S79]:
    Path: ['ne99', 'ne61']
    Switches: ['D05_R']
route_36 [S55 >> S96]:
    Path: ['ne99', 'ne63']
    Switches: ['D05_N']
route_37 [S59 >> S111]:
    Path: ['ne101', 'ne912']
    Switches: ['D09_N']
route_38 [S59 >> S82]:
    Path: ['ne101', 'ne910']
    Switches: ['D09_R']
route_39 [S64 >> S69]:
    Path: ['ne991', 'ne288']
    Switches: ['D01_R']
route_40 [S64 >> S87]:
    Path: ['ne991', 'ne290']
    Switches: ['D01_N']
route_41 [C67 << T01]:
    Path: ['ne295', 'ne288', 'ne991']
    Switches: ['D01_R', 'D03_N']
route_42 [B68 >> S124]:
    Path: ['ne110', 'ne384', 'ne297']
    Switches: ['D04_R']
route_43 [S69 >> C70]:
    Path: ['ne288', 'ne295']
    Switches: ['D03_N']
route_44 [S69 >> B68]:
    Path: ['ne288', 'ne110']
    Switches: ['D03_R']
route_45 [C70 >> S124]:
    Path: ['ne295', 'ne297']
    Switches: ['D04_N']
route_46 [B71 << T13]:
    Path: ['ne384', 'ne104']
route_47 [B71 << T01]:
    Path: ['ne384', 'ne110', 'ne288', 'ne991']
    Switches: ['D01_R', 'D03_R']
route_48 [S72 << C67]:
    Path: ['ne297', 'ne295']
    Switches: ['D04_N']
route_49 [S72 << B71]:
    Path: ['ne297', 'ne384']
    Switches: ['D04_R']
route_50 [C74 >> S84]:
    Path: ['ne404', 'ne400']
    Switches: ['D07_N']
route_51 [S75 << C77]:
    Path: ['ne400', 'ne404']
    Switches: ['D07_N']
route_52 [S75 << S94]:
    Path: ['ne400', 'ne407']
    Switches: ['D07_R']
route_53 [C77 << X51]:
    Path: ['ne404', 'ne61', 'ne99', 'ne98']

```

```

    Switches: ['D05_R', 'D08_N']
route_54 [B78 >> S84]:
    Path: ['ne123', 'ne407', 'ne400']
    Switches: ['D07_R', 'D16_N']
route_55 [S79 >> C74]:
    Path: ['ne61', 'ne404']
    Switches: ['D08_N']
route_56 [S79 >> B78]:
    Path: ['ne61', 'ne123']
    Switches: ['D08_R']
route_57 [C80 << J43]:
    Path: ['ne130', 'ne910', 'ne101']
    Switches: ['D09_R', 'D10_N']
route_58 [B81 >> S105]:
    Path: ['ne135', 'ne127']
    Switches: ['D24_N']
route_59 [S82 >> S108]:
    Path: ['ne910', 'ne130']
    Switches: ['D10_N']
route_60 [S82 >> B81]:
    Path: ['ne910', 'ne135']
    Switches: ['D10_R']
route_61 [C83 << S75]:
    Path: ['ne993', 'ne400']
    Switches: ['Sw02_N']
route_62 [S84 >> T25]:
    Path: ['ne400', 'ne993']
    Switches: ['Sw02_N']
route_63 [S84 >> J40]:
    Path: ['ne400', 'ne994', 'ne992']
    Switches: ['Sw02_R', 'Sw01_R']
route_64 [S87 >> J36]:
    Path: ['ne290', 'ne292']
    Switches: ['D14_N']
route_65 [S87 >> T03]:
    Path: ['ne290', 'ne111', 'ne377']
    Switches: ['D14_R', 'D15_R']
route_66 [S94 << T17]:
    Path: ['ne407', 'ne122']
    Switches: ['D16_R']
route_67 [S94 << X51]:
    Path: ['ne407', 'ne123', 'ne61', 'ne99', 'ne98']
    Switches: ['D05_R', 'D08_R', 'D16_N']
route_68 [C95 << C54]:
    Path: ['ne65', 'ne63']
    Switches: ['D18_N']
route_69 [S96 >> C120]:
    Path: ['ne63', 'ne65']
    Switches: ['D18_N']
route_70 [S96 >> T05]:
    Path: ['ne63', 'ne124', 'ne421']
    Switches: ['D18_R', 'D19_R']
route_71 [S102 << J47]:
    Path: ['ne114', 'ne132']
    Switches: ['D23_N']
route_72 [S102 << S118]:
    Path: ['ne114', 'ne129', 'ne127']
    Switches: ['D23_R', 'D12_RR']
route_73 [S102 << C80]:
    Path: ['ne114', 'ne129', 'ne130']
    Switches: ['D23_R', 'D12_NR']
route_74 [S105 >> L29]:
    Path: ['ne127', 'ne129', 'ne114']
    Switches: ['D23_R', 'D12_RR']

```

```

route_75 [S105 >> T07]:
    Path: ['ne127', 'ne450']
    Switches: ['D12_RN']
route_76 [S108 >> L29]:
    Path: ['ne130', 'ne129', 'ne114']
    Switches: ['D23_R', 'D12_NR']
route_77 [S108 >> T07]:
    Path: ['ne130', 'ne450']
    Switches: ['D12_NN']
route_78 [S111 >> T09]:
    Path: ['ne912', 'ne131', 'ne465']
    Switches: ['D20_R', 'D21_R']
route_79 [S111 >> J46]:
    Path: ['ne912', 'ne132']
    Switches: ['D20_N']
route_80 [S118 << J43]:
    Path: ['ne127', 'ne135', 'ne910', 'ne101']
    Switches: ['D09_R', 'D10_R', 'D24_N']
route_81 [S118 << T23]:
    Path: ['ne127', 'ne134']
    Switches: ['D24_R']
route_82 [C120 >> J40]:
    Path: ['ne65', 'ne992']
    Switches: ['Sw01_N']
route_83 [S121 << C95]:
    Path: ['ne992', 'ne65']
    Switches: ['Sw01_N']
route_84 [S121 << S75]:
    Path: ['ne992', 'ne994', 'ne400']
    Switches: ['Sw02_R', 'Sw01_R']
route_85 [C123 << S72]:
    Path: ['ne995', 'ne297']
    Switches: ['Sw03_N']
route_86 [S124 >> T27]:
    Path: ['ne297', 'ne995']
    Switches: ['Sw03_N']
route_87 [S124 >> J38]:
    Path: ['ne297', 'ne997', 'ne996']
    Switches: ['Sw03_R', 'Sw04_R']
route_88 [S127 << J37]:
    Path: ['ne996', 'ne292']
    Switches: ['Sw04_N']
route_89 [S127 << S72]:
    Path: ['ne996', 'ne997', 'ne297']
    Switches: ['Sw03_R', 'Sw04_R']

```

C.4. Red de grafos generada por el RNA

La información exportada en el Código C.3 (Infrastructure.RNA) Código C.4 (SafePoint.RNA), Código C.6 (Signalling.RNA) y Código C.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura C.8.

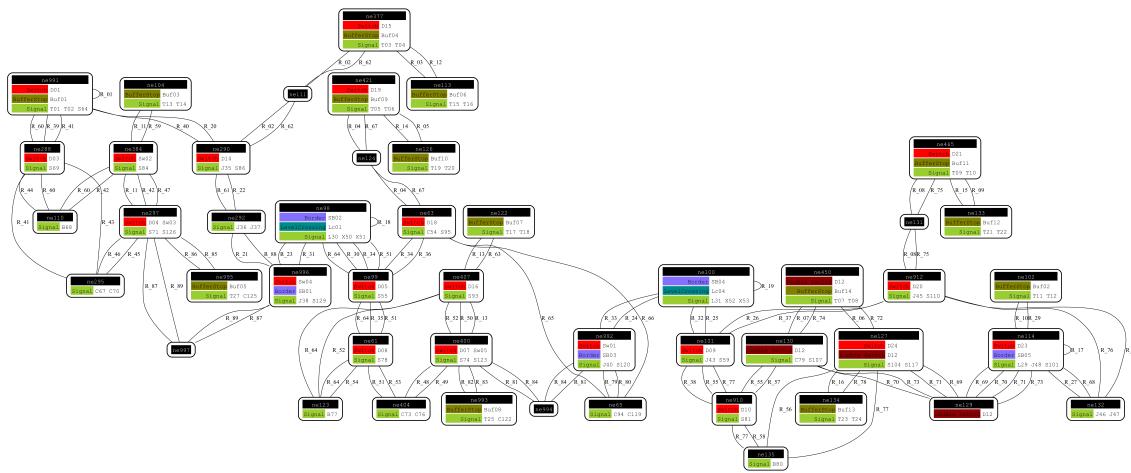


Figura C.8: Red de grafos generada por el RNA para el ejemplo 4.

Cada nodo del grafo de la Figura C.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

C.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código C.8 en una hoja de cálculo, similar a la que se visualiza en las Tablas C.6, C.7, C.8, C.9, C.10 y C.11.

Tabla C.6: Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 1 a 15).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	S ₇₁				ne991-ne991
R ₀₂	T ₀₄	J ₃₅	D ₁₄ ^R -D ₁₅ ^R			ne377-ne290
R ₀₃	T ₀₄	T ₁₅	D ₁₅ ^N			ne377-ne113
R ₀₄	T ₀₆	C ₆₂	D ₁₈ ^R -D ₁₉ ^R			ne421-ne63
R ₀₅	T ₀₆	T ₁₉	D ₁₉ ^N			ne421-ne126
R ₀₆	T ₀₈	S ₁₁₉				ne450-ne127
R ₀₇	T ₀₈	C ₈₇				ne450-ne130
R ₀₈	T ₁₀	J ₄₅	D ₂₀ ^R -D ₂₁ ^R			ne465-ne912
R ₀₉	T ₁₀	T ₂₁	D ₂₁ ^N			ne465-ne133
R ₁₀	T ₁₂	S ₁₀₉				ne102-ne114
R ₁₁	T ₁₄	S ₁₂₅	D ₀₄ ^R			ne104-ne297
R ₁₂	T ₁₆	T ₀₃	D ₁₅ ^N			ne113-ne377
R ₁₃	T ₁₈	S ₉₁	D ₀₇ ^R -D ₁₆ ^R			ne122-ne400
R ₁₄	T ₂₀	T ₀₅	D ₁₉ ^N			ne126-ne421
R ₁₅	T ₂₂	T ₀₉	D ₂₁ ^N			ne133-ne465

Tabla C.7: Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 16 a 30).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₁₆	T ₂₄	S ₁₀₅	D ₂₄ ^R			ne134-ne127
R ₁₇	L ₂₉	J ₄₈				ne114-ne114
R ₁₈	S ₃₀	X ₅₀				ne98-ne98
R ₁₉	L ₃₁	X ₅₂				ne100-ne100
R ₂₀	J ₃₅	T ₀₁	D ₀₁ ^N			ne290-ne991
R ₂₁	J ₃₆	J ₃₈	Sw ₀₄ ^N			ne292-ne996
R ₂₂	J ₃₇	J ₃₅	D ₁₄ ^N			ne292-ne290
R ₂₃	J ₃₈	S ₃₀				ne996-ne98
R ₂₄	J ₄₀	L ₃₁				ne992-ne100
R ₂₅	J ₄₃	X ₅₃				ne101-ne100
R ₂₆	J ₄₅	J ₄₃	D ₀₉ ^N			ne912-ne101
R ₂₇	J ₄₆	L ₂₉	D ₂₃ ^N			ne132-ne114
R ₂₈	J ₄₇	J ₄₅	D ₂₀ ^N			ne132-ne912
R ₂₉	J ₄₈	T ₁₁				ne114-ne102
R ₃₀	X ₅₀	S ₅₅		Lc01		ne98-ne99

Tabla C.8: Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 31 a 45).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₃₁	X ₅₁	S ₁₂₇			Lc ₀₁	ne98-ne996
R ₃₂	X ₅₂	S ₅₉			Lc ₀₄	ne100-ne101
R ₃₃	X ₅₃	S ₁₂₁			Lc ₀₄	ne100-ne992
R ₃₄	C ₅₄	X ₅₁	D ₀₅ ^N			ne63-ne98
R ₃₅	S ₅₅	S ₇₉	D ₀₅ ^R			ne99-ne61
R ₃₆	S ₅₅	S ₉₆	D ₀₅ ^N			ne99-ne63
R ₃₇	S ₅₉	S ₁₁₁	D ₀₉ ^N			ne101-ne912
R ₃₈	S ₅₉	S ₈₂	D ₀₉ ^R			ne101-ne910
R ₃₉	S ₆₄	S ₆₉	D ₀₁ ^R			ne991-ne288
R ₄₀	S ₆₄	S ₈₇	D ₀₁ ^N			ne991-ne290
R ₄₁	C ₆₇	T ₀₁	D ₀₁ ^R -D ₀₃ ^N			ne295-ne991
R ₄₂	B ₆₈	S ₁₂₄	D ₀₄ ^R			ne110-ne297
R ₄₃	S ₆₉	C ₇₀	D ₀₃ ^N			ne288-ne295
R ₄₄	S ₆₉	B ₆₈	D ₀₃ ^R			ne288-ne110
R ₄₅	C ₇₀	S ₁₂₄	D ₀₄ ^N			ne295-ne297

Tabla C.9: Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 46 a 60).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₄₆	B ₇₁	T ₁₃				ne384-ne104
R ₄₇	B ₇₁	T ₀₁	D ₀₁ ^R -D ₀₃ ^R			ne384-ne991
R ₄₈	S ₇₂	C ₆₇	D ₀₄ ^N			ne297-ne295
R ₄₉	S ₇₂	B ₇₁	D ₀₄ ^R			ne297-ne384
R ₅₀	C ₇₄	S ₈₄	D ₀₇ ^N			ne404-ne400
R ₅₁	S ₇₅	C ₇₇	D ₀₇ ^N			ne400-ne404
R ₅₂	S ₇₅	S ₉₄	D ₀₇ ^R			ne400-ne407
R ₅₃	C ₇₇	X ₅₁	D ₀₅ ^R -D ₀₈ ^N			ne404-ne98
R ₅₄	B ₇₈	S ₈₄	D ₀₇ ^R -D ₁₆ ^N			ne123-ne400
R ₅₅	S ₇₉	C ₇₄	D ₀₈ ^N			ne61-ne404
R ₅₆	S ₇₉	B ₇₈	D ₀₈ ^R			ne61-ne123
R ₅₇	C ₈₀	J ₄₃	D ₀₉ ^R -D ₁₀ ^N			ne130-ne101
R ₅₈	B ₈₁	S ₁₀₅	D ₂₄ ^N			ne135-ne127
R ₅₉	S ₈₂	S ₁₀₈	D ₁₀ ^N			ne910-ne130
R ₆₀	S ₈₂	B ₈₁	D ₁₀ ^R			ne910-ne135

Tabla C.10: Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 61 a 75).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₆₁	C ₈₃	S ₇₅	Sw ₀₂ ^N			ne993-ne400
R ₆₂	S ₈₄	T ₂₅	Sw ₀₂ ^N			ne400-ne993
R ₆₃	S ₈₄	J ₄₀	Sw ₀₂ ^R -Sw ₀₁ ^R			ne400-ne992
R ₆₄	S ₈₇	J ₃₆	D ₁₄ ^N			ne290-ne292
R ₆₅	S ₈₇	T ₀₃	D ₁₄ ^R -D ₁₅ ^R			ne290-ne377
R ₆₆	S ₉₄	T ₁₇	D ₁₆ ^R			ne407-ne122
R ₆₇	S ₉₄	X ₅₁	D ₀₅ ^R -D ₀₈ ^R -D ₁₆ ^N			ne407-ne98
R ₆₈	C ₉₅	C ₅₄	D ₁₈ ^N			ne65-ne63
R ₆₉	S ₉₆	C ₁₂₀	D ₁₈ ^N			ne63-ne65
R ₇₀	S ₉₆	T ₀₅	D ₁₈ ^R -D ₁₉ ^R			ne63-ne421
R ₇₁	S ₁₀₂	J ₄₇	D ₂₃ ^N			ne114-ne132
R ₇₂	S ₁₀₂	S ₁₁₈	D ₂₃ ^R -D ₁₂ ^{RR}			ne114-ne127
R ₇₃	S ₁₀₂	C ₈₀	D ₂₃ ^R -D ₁₂ ^{NR}			ne114-ne130
R ₇₄	S ₁₀₅	L ₂₉	D ₂₃ ^R -D ₁₂ ^{RR}			ne912-ne465
R ₇₅	S ₁₀₅	T ₀₇	D ₁₂ ^{RN}			ne912-ne132

Tabla C.11: Tabla de enclavamiento del ejemplo 4 generada por el RNA (Rutas 76 a 89).

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₇₆	S ₁₀₈	L ₂₉	D ₂₃ ^R -D ₁₂ ^{NR}			ne130-ne114
R ₇₇	S ₁₀₈	T ₀₇	D ₁₂ ^{NN}			ne130-ne450
R ₇₈	S ₁₁₁	T ₀₉	D ₂₀ ^R -D ₂₁ ^R			ne912-ne465
R ₇₉	S ₁₁₁	J ₄₆	D ₂₀ ^N			ne912-ne132
R ₈₀	S ₁₁₈	J ₄₃	D ₀₉ ^R -D ₁₀ ^R -D ₂₄ ^N			ne127-ne101
R ₈₁	S ₁₁₈	T ₂₃	D ₂₄ ^R			ne127-ne134
R ₈₂	C ₁₂₀	J ₄₀	Sw ₀₁ ^N			ne65-ne992
R ₈₃	S ₁₂₁	C ₉₅	Sw ₀₁ ^N			ne992-ne65
R ₈₄	S ₁₂₁	S ₇₅	Sw ₀₂ ^R -Sw ₀₁ ^R			ne992-ne400
R ₈₅	C ₁₂₃	S ₇₂	Sw ₀₃ ^N			ne995-ne297
R ₈₆	S ₁₂₄	T ₂₇	Sw ₀₃ ^N			ne297-ne995
R ₈₇	S ₁₂₄	J ₃₈	Sw ₀₃ ^R -Sw ₀₄ ^R			ne297-ne996
R ₈₈	S ₁₂₈	J ₃₇	Sw ₀₄ ^N			ne996-ne292
R ₈₉	S ₁₂₈	S ₇₂	Sw ₀₃ ^R -Sw ₀₄ ^R			ne996-ne297

En una primera inspección podemos ver que el nuevo señalamiento tiene 89 rutas, versus las 77 rutas del señalamiento original (ver Tabla ??). Esto se debe a que muchas de las rutas generadas son particiones de las rutas originales, al ser demasiado extensas o abarcar demasiados elementos ferroviarios sin paradas intermedias.

C.6. Validación del sistema de enclavamientos

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 77 rutas del señalamiento original (Tabla C.1, Tabla C.2, Tabla C.3, Tabla C.4 y Tabla C.5) tienen 77 rutas equivalentes en el señalamiento generado por el RNA (Tabla C.6, Tabla C.7, Tabla C.8, Tabla C.9, Tabla C.10 y Tabla C.11), tal como se puede visualizar en las Tabla C.12, C.13, C.14, C.15 y C.16, generadas automáticamente por el RNA.

Tabla C.12: Equivalencias entre las rutas 1 a 15 originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S ₁₅ -S ₂₄	R ₃₀	X ₅₀ -S ₅₅
R ₀₂	S ₁₆ -S ₆₇	R ₃₁ +R ₈₈	X ₅₃ -J ₃₇
R ₀₃	S ₁₆ -S ₄₃	R ₃₁ +R ₈₉	X ₅₃ -S ₇₂
R ₀₄	S ₁₉ -S ₃₂	R ₃₂	X ₅₂ -S ₅₉
R ₀₅	S ₂₀ -S ₁₁	R ₃₃ +R ₈₃	X ₅₃ -C ₉₅
R ₀₆	S ₂₀ -S ₅₆	R ₃₃ +R ₈₄	X ₅₃ -S ₇₅
R ₀₇	S ₂₃ -S ₁₆	R ₃₄	C ₅₄ -X ₅₁
R ₀₈	S ₂₄ -S ₈₀	R ₃₆	S ₅₅ -S ₉₆
R ₀₉	S ₂₄ -S ₅₂	R ₃₅ +R ₅₅	S ₅₅ -C ₇₄
R ₁₀	S ₂₄ -S ₅₇	R ₃₅ +R ₅₆	S ₅₅ -B ₇₈
R ₁₁	S ₂₇ -S ₅₆	R ₆₁	C ₈₃ -S ₇₅
R ₁₂	S ₂₈ -S ₀₄	R ₅₁ +R ₅₃ +R ₃₁ +R ₈₉ +R ₄₉	S ₇₅ -B ₇₁
R ₁₃	S ₂₈ -S ₁₉	R ₆₃ +R ₂₄	S ₈₄ -L ₃₁
R ₁₄	S ₃₁ -S ₂₀	R ₂₆ +R ₂₅	J ₄₅ -X ₅₃
R ₁₅	S ₃₂ -S ₉₁	R ₃₇	S ₅₉ -S ₁₁₁

Tabla C.13: Equivalencias entre las rutas 16 a 30 originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₁₆	S ₃₂ -S ₅₉	R ₃₈ +R ₅₉	S ₅₉ -S ₁₀₈
R ₁₇	S ₃₂ -S ₆₀	R ₃₈ +R ₆₀	S ₅₉ -B ₈₁
R ₁₈	S ₃₅ -S ₀₁	R ₆₄ +R ₂₁ +R ₂₃ +R ₃₀ +R ₃₆ +R ₆₉ +R ₈₂ +R ₂₄ +R ₃₂ +R ₃₈ +R ₅₉	S ₈₇ -S ₁₀₈
R ₁₉	S ₃₉ -S ₄₃	R ₈₅	C ₁₂₃ -S ₇₂
R ₂₀	S ₄₄ -S ₀₄	R ₄₂ +R ₄₉	B ₆₈ -B ₇₁
R ₂₁	S ₄₅ -S ₀₁	R ₄₅ +R ₈₇ +R ₂₃ +R ₃₀ +R ₃₆ +R ₆₉ +R ₈₂ +R ₂₄ +R ₃₂ +R ₃₈ +R ₅₉	C ₇₀ -S ₁₀₈
R ₂₂	S ₄₆ -S ₆₃	R ₄₇ +R ₃₉ +R ₄₄	B ₇₁ -B ₆₈
R ₂₃	S ₄₆ -S ₀₉	R ₄₇ +R ₄₀ +R ₆₄	B ₇₁ -J ₃₆
R ₂₄	S ₄₇ -S ₄₀	R ₄₅	C ₇₀ -S ₁₂₄
R ₂₅	S ₅₂ -S ₂₈	R ₅₀	C ₇₄ -S ₈₄
R ₂₆	S ₅₆ -S ₅₈	R ₅₁	S ₇₅ -C ₇₇
R ₂₇	S ₅₆ -S ₀₃	R ₅₂	S ₇₅ -S ₉₄
R ₂₈	S ₅₇ -S ₇₆	R ₅₄ +R ₅₂	B ₇₈ -S ₉₄
R ₂₉	S ₅₈ -S ₁₆	R ₅₃	C ₇₇ -X ₅₁
R ₃₀	S ₅₉ -S ₈₇	R ₇₆ +R ₄₁ +R ₆₃	S ₁₀₈ -L ₂₉

Tabla C.14: Equivalencias entre las rutas 31 a 45 originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₃₁	S ₅₉ -S ₀₆	R ₅₇ +R ₂₅ +R ₃₃ +R ₈₃ +R ₆₈ +R ₃₄ +R ₃₁ +R ₈₈ +R ₂₃ +R ₆₅	C ₈₀ -T ₀₃
R ₃₂	S ₆₀ -S ₉₉	R ₅₈	B ₈₁ -S ₁₀₅
R ₃₃	S ₆₂ -S ₀₄	R ₁₁ +R ₄₉	T ₁₄ -B ₇₁
R ₃₄	S ₆₃ -S ₀₁	R ₄₂ +R ₈₇ +R ₂₃ +R ₃₀ +R ₃₆ +R ₆₉ +R ₈₂ +R ₂₄ +R ₃₂ +R ₃₈ +R ₅₉	B ₆₈ -S ₁₀₈
R ₃₅	S ₆₇ -S ₀₉	R ₈₈	S ₁₂₈ -J ₃₇
R ₃₆	S ₆₈ -S ₀₈	R ₄₀	S ₆₄ -S ₈₇
R ₃₇	S ₇₄ -S ₇₆	R ₁₃ +R ₅₂	T ₁₈ -S ₉₄
R ₃₈	S ₇₅ -S ₁₆	R ₅₄ +R ₅₁ +R ₅₃	S ₈₅ -S ₅₃
R ₃₉	S ₇₆ -S ₂₈	R ₆₆ +R ₁₃	B ₇₈ -X ₅₁
R ₄₀	S ₈₀ -S ₁₀	R ₃₀ +R ₃₆	X ₅₀ -S ₉₆
R ₄₁	S ₈₃ -S ₁₂	R ₀₅	T ₀₆ -T ₁₉
R ₄₂	S ₈₄ -S ₈₅	R ₇₀	S ₉₆ -T ₀₅
R ₄₃	S ₈₅ -S ₀₅	R ₇₀	S ₉₆ -T ₀₅
R ₄₄	S ₈₆ -S ₈₇	R ₂₇	J ₄₆ -L ₂₉
R ₄₅	S ₈₇ -S ₀₈	R ₇₂ +R ₈₀ +R ₂₅ +R ₃₃ +R ₈₃ +R ₆₈ +R ₃₄ +R ₃₁ +R ₈₈ +R ₂₂	S ₁₀₂ -J ₃₅

Tabla C.15: Equivalencias entre las rutas 46 a 60 originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₄₆	S ₈₉ -S ₃₁	R ₂₈	J ₄₇ -J ₄₅
R ₄₇	S ₉₀ -S ₈₉	R ₇₁	S ₁₀₂ -J ₄₇
R ₄₈	S ₉₁ -S ₉₃	R ₃₇	S ₅₉ -S ₁₁₁
R ₄₉	S ₉₃ -S ₈₆	R ₇₉	S ₁₁₁ -J ₄₆
R ₅₀	S ₉₃ -S ₉₅	R ₇₈	S ₁₁₁ -T ₀₉
R ₅₁	S ₉₄ -S ₁₃	R ₀₉	T ₁₀ -T ₂₁
R ₅₂	S ₉₅ -S ₉₆	R ₇₈	S ₁₁₁ -T ₀₉
R ₅₃	S ₉₆ -S ₀₇	R ₀₈ +R ₂₆ +R ₂₅ +R ₃₃ +R ₈₃ +R ₆₈ +R ₃₄ +R ₃₁ +R ₈₈ +R ₂₂ +R ₂₀	T ₁₀ -T ₀₁
R ₅₄	S ₉₇ -S ₉₉	R ₁₆ +R ₇₇	T ₂₄ -S ₁₀₅
R ₅₅	S ₉₈ -S ₂₀	R ₅₈ +R ₈₀ +R ₂₅	B ₈₁ -x ₅₃
R ₅₆	S ₉₉ -S ₈₇	R ₇₄	S ₁₀₅ -L ₂₉
R ₅₇	S ₉₉ -S ₀₆	R ₈₀ +R ₂₅ +R ₃₃ +R ₈₃ +R ₆₈ +R ₃₄ +R ₃₁ +R ₈₈ +R ₂₂ +R ₆₅	S ₁₁₈ -T ₀₃
R ₅₈	S ₄₀ -S ₀₂	R ₈₇ +R ₂₃ +R ₃₀ +R ₃₆ +R ₆₉	S ₁₂₄ -C ₁₂₀
R ₅₉	S ₄₀ -S ₁₅	R ₈₇ +R ₂₃	S ₁₂₄ -S ₃₀
R ₆₀	S ₄₃ -S ₄₅	R ₄₈	S ₇₂ -C ₆₇

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Tabla C.16: Equivalencias entre las rutas 61 a 77 originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₆₁	S ₄₃ -S ₄₆	R ₄₉	S ₇₂ -B ₇₁
R ₆₂	S ₀₁ -S ₂₀	R ₅₇ +R ₂₅	C ₈₀ -X ₅₃
R ₆₃	S ₀₂ -S ₁₉	R ₈₂ +R ₂₄	C ₁₂₀ -L ₃₁
R ₆₄	S ₀₃ -S ₇₅	R ₆₇ +R ₃₀ +R ₃₅ +R ₅₆	S ₉₄ -B ₇₈
R ₆₅	S ₀₃ -S ₁₁	R ₆₆ +R ₁₃ +R ₆₃ +R ₈₃	S ₉₄ -C ₉₅
R ₆₆	S ₀₄ -S ₄₀	R ₄₆ +R ₁₁	B ₇₁ -S ₁₂₅
R ₆₇	S ₀₆ -S ₁₀	R ₀₂ +R ₆₄ +R ₂₁ +R ₂₃ +R ₃₀ +R ₃₆	T ₀₄ -S ₉₆
R ₆₈	S ₀₆ -S ₃₅	R ₀₂	T ₀₄ -J ₃₅
R ₆₉	S ₀₇ -S ₆₈	R ₄₀	S ₆₄ -S ₈₇
R ₇₀	S ₀₇ -S ₄₇	R ₃₉ +R ₄₃	S ₆₄ -C ₇₀
R ₇₁	S ₀₇ -S ₄₄	R ₃₉ +R ₄₄	S ₆₄ -B ₆₈
R ₇₂	S ₀₈ -S ₁₅	R ₆₄ +R ₂₁ +R ₂₃	S ₈₇ -S ₃₀
R ₇₃	S ₀₈ -S ₀₃	R ₆₄ +R ₂₁ +R ₂₃ +R ₃₀ +R ₃₅ +R ₅₅ +R ₅₀ +R ₅₂	S ₈₇ -S ₉₄
R ₇₄	S ₀₉ -S ₃₅	R ₂₂	J ₃₇ -J ₃₅
R ₇₅	S ₁₀ -S ₀₂	R ₆₉	S ₉₆ -C ₁₂₀
R ₇₆	S ₁₀ -S ₈₄	R ₇₀	S ₉₆ -T ₀₅
R ₇₇	S ₁₁ -S ₂₃	R ₆₈	C ₉₅ -C ₅₄

De las 89 rutas generadas, 35 rutas tienen una equivalencia directa con alguna de las 77 rutas originales. Un total de 36 de las nuevas rutas son producto de la partición de alguna de las rutas originales. Las restantes 18 rutas generadas (R₁, R₃, R₄, R₆, R₇, R₁₀, R₁₂, R₁₄, R₁₅, R₁₇, R₁₈, R₁₉, R₂₉, R₆₂, R₇₅, R₇₇, R₈₁ y R₈₆) no tienen ninguna equivalencia con el señalamiento original, y todas ellas son producto de haber añadido señales de parada o de partida para a los finales de vía absolutos, ausentes en el señalamiento original.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

C.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura C.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código C.8.

Código C.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:47
n_switch:21
n_doubleSwitch:1
n_borders:5
n_buffers:14
n_levelCrossings:2
n_platforms:0
n_scissorCrossings:0
n_signals:77
N : 238
```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 256 archivos en formato VHDL. El archivo *Arty.Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 4 para automatizar la importación de los archivos del ejemplo 4 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 238 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 238 en el Código C.8. El ejemplo 4 utiliza mas de 82000 sub módulos conectados automáticamente mediante mas de 162000 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla C.17. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es mediana y el tiempo de síntesis e implementación es de 2 minutos con 2 segundos y 3 minutos con 13 segundos, respectivamente.

Tabla C.17: Síntesis e implementación del ejemplo 4 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	13801	13666	25.94-25.69 %
FF	15069	15072	14.16-14.17 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice D

Ejemplo 5

D.1. Topología ferroviaria original

El quinto ejemplo, ilustrado en la Figura D.1, es una topología de dos vías principales paralelas y desconexas, con sus correspondientes desvíos para poder operar cada línea en ambos sentidos. La primera vía principal, la vía superior, utiliza los cambios de vías Sw01 y Sw02 para permitir circular dos formaciones en sentido contrario. En tanto que la segunda vía principal, la vía inferior, utiliza los cambios de vías Sw03 y Sw04 para el mismo fin. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología de dos bypasses independientes y desconexos.

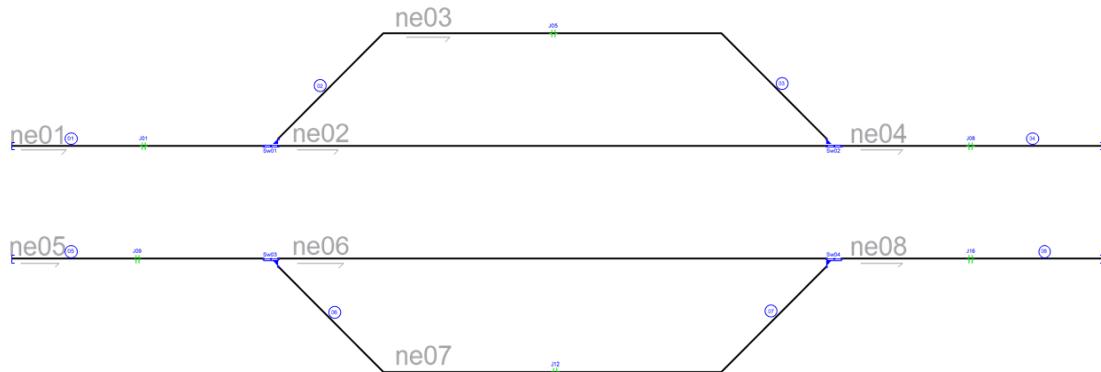


Figura D.1: Topología ferroviaria del ejemplo 5 sin señalamiento.

Para incrementar la dificultad del análisis y obtener resultados mas completos, todos los finales de vías absolutos. Además, se incluyeron junturas entre los finales de vías y los cambios de vías.

D.2. Señalamiento original

El señalamiento original, ilustrado en la Figura D.2, incluye señales de parada próximas a los finales de vías absolutas (S13, S14, S15, S16), señales de maniobras antes de converger en una vía principal (S03, S04, S05, S06, S08, S09, S10, S11) y señales múltiples para cambios de vías divergentes (S01, S02, S07, S12).

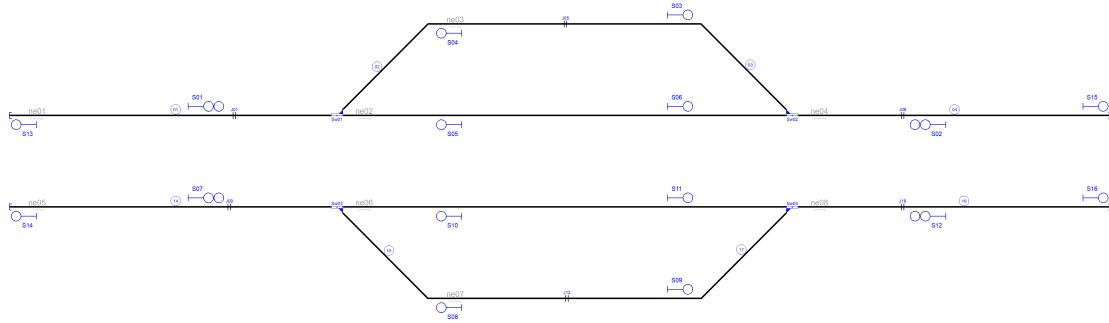


Figura D.2: Señalamiento original del ejemplo 5.

Estas señales permiten definir hasta un máximo de 16 rutas, todas ellas detalladas en la Tabla D.1. En una primera inspección, se puede comprobar que todos los elementos ferroviarios son alcanzados por al menos una de las rutas, en al menos una dirección. Además, todos los cambios de vías son utilizados, de forma simple o compuesta.

Tabla D.1: Tabla de enclavamiento original del ejemplo 5.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	S ₀₁	S ₀₆	Sw ₀₁ ^N	-	-	ne ₀₁ -ne ₀₂
R ₀₂	S ₀₅	S ₁₃	Sw ₀₁ ^N	-	-	ne ₀₂ -ne ₀₁
R ₀₃	S ₀₁	S ₀₃	Sw ₀₁ ^R	-	-	ne ₀₁ -ne ₀₃
R ₀₄	S ₀₄	S ₁₃	Sw ₀₁ ^R	-	-	ne ₀₃ -ne ₀₁
R ₀₅	S ₀₂	S ₀₄	Sw ₀₂ ^R	-	-	ne ₀₄ -ne ₀₂
R ₀₆	S ₀₆	S ₁₅	Sw ₀₂ ^N	-	-	ne ₀₂ -ne ₀₄
R ₀₇	S ₀₂	S ₀₅	Sw ₀₂ ^N	-	-	ne ₀₄ -ne ₀₃
R ₀₈	S ₀₃	S ₁₅	Sw ₀₂ ^R	-	-	ne ₀₃ -ne ₀₄
R ₀₉	S ₀₇	S ₁₁	Sw ₀₃ ^N	-	-	ne ₀₅ -ne ₀₆
R ₁₀	S ₁₀	S ₁₄	Sw ₀₃ ^N	-	-	ne ₀₆ -ne ₀₅
R ₁₁	S ₀₇	S ₀₉	Sw ₀₃ ^R	-	-	ne ₀₅ -ne ₀₇
R ₁₂	S ₀₈	S ₁₄	Sw ₀₃ ^R	-	-	ne ₀₇ -ne ₀₅
R ₁₃	S ₁₂	S ₁₀	Sw ₀₄ ^N	-	-	ne ₀₈ -ne ₀₆
R ₁₄	S ₁₁	S ₁₆	Sw ₀₄ ^N	-	-	ne ₀₆ -ne ₀₈
R ₁₅	S ₁₂	S ₀₈	Sw ₀₄ ^R	-	-	ne ₀₈ -ne ₀₇
R ₁₆	S ₀₉	S ₁₆	Sw ₀₄ ^R	-	-	ne ₀₇ -ne ₀₈

Todas las rutas abarcan mas de un *netElement*, como por ejemplo la ruta R16 que comienza en la señal S09 y finaliza en la señal S16, atravesando los *netElements* ne07 y ne08, utilizando el cambio de vías Sw04 en posición reversa.

D.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código D.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código D.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analyzing railML object
Analyzing graph
ne01 [-1451, -150] [-763, -150] >>
ne02 [-763, -150] [736, -150] >>
ne03 [-763, -150] [736, -150] >>
ne04 [736, -150] [1451, -150] >>
ne05 [-1451, -450] [-763, -450] >>
ne06 [-763, -450] [736, -450] >>
ne07 [-763, -450] [736, -450] >>
ne08 [736, -450] [1451, -450] >>
The network is not connected
```

Por ejemplo, el *netElement* ne08 inicia en la coordenada (736;-450) y finaliza en la coordenada (1451;-450). El símbolo >> indica que ne08 se encuentra definido de izquierda a derecha, ya que la componente x de la coordenada final es mayor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura D.2. Por ejemplo, ne01, ne02 y ne03 comparten la coordenada (736;-450), que coincide con la coordenada del cambio de vías Sw01.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código D.2.

Código D.2: Detección de puntos críticos por parte del RNA

```
Analyzing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne01 has a RailJoint[J01] @ [-1101, -150]
ne02 has a Middle point @ [-548.9, -150]
ne02 has a Middle point @ [-334.7, -150]
ne02 has a Middle point @ [-120.6, -150]
ne02 has a Middle point @ [93.6, -150]
ne02 has a Middle point @ [307.7, -150]
ne02 has a Middle point @ [521.9, -150]
ne03 has a RailJoint[J05] @ [-11, 150]
ne03 has a Curve(3 lines) @ [[-463, 150], [436, 150]]
```

```

ne04 has a RailJoint[J08] @ [1100, -150]
ne05 has a RailJoint[J09] @ [-1118, -450]
ne06 has a Middle point @ [-548.9, -450]
ne06 has a Middle point @ [-334.7, -450]
ne06 has a Middle point @ [-120.6, -450]
ne06 has a Middle point @ [93.6, -450]
ne06 has a Middle point @ [307.7, -450]
ne06 has a Middle point @ [521.9, -450]
ne07 has a RailJoint[J12] @ [-5, -750]
ne07 has a Curve(3 lines) @ [[-463, -750], [436, -750]]
ne08 has a RailJoint[J16] @ [1100, -450]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código D.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código D.3: Infrastructure.RNA

```

Nodes: 8|Switches: 4|Signals: 0|Detectors: 6|Ends: 4|Barriers: 0
Node ne01:
    Track = track1
    TrainDetectionElements -> tde89
        Type -> insulatedRailJoint
        Type = BufferStop -> ['bus174']
    Neighbours = 2 -> ['ne02', 'ne03']
    Switches -> Sw01
        ContinueCourse -> right -> ne02
        BranchCourse -> left -> ne03
Node ne02:
    Track = track5
    Neighbours = 3 -> ['ne01', 'ne03', 'ne04']
Node ne03:
    Track = track6
    TrainDetectionElements -> tde93
        Type -> insulatedRailJoint
    Neighbours = 3 -> ['ne01', 'ne02', 'ne04']
Node ne04:
    Track = track3
    TrainDetectionElements -> tde96
        Type -> insulatedRailJoint
        Type = BufferStop -> ['bus176']
    Neighbours = 2 -> ['ne02', 'ne03']
    Switches -> Sw02
        ContinueCourse -> left -> ne02
        BranchCourse -> right -> ne03
Node ne05:
    Track = track2
    TrainDetectionElements -> tde105
        Type -> insulatedRailJoint
        Type = BufferStop -> ['bus175']
    Neighbours = 2 -> ['ne06', 'ne07']
    Switches -> Sw03

```

```

        ContinueCourse -> left -> ne06
        BranchCourse -> right -> ne07

Node ne06:
    Track = track7
    Neighbours = 3 -> ['ne05', 'ne07', 'ne08']

Node ne07:
    Track = track8
    TrainDetectionElements -> tde108
        Type -> insulatedRailJoint
    Neighbours = 3 -> ['ne05', 'ne06', 'ne08']

Node ne08:
    Track = track4
    TrainDetectionElements -> tde112
        Type -> insulatedRailJoint
        Type = BufferStop -> ['bus177']
    Neighbours = 2 -> ['ne06', 'ne07']
    Switches -> Sw04
        ContinueCourse -> right -> ne06
        BranchCourse -> left -> ne07

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código D.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código D.4: SafePoints.RNA

```

ne01:
    Next: [[-1201.0, -150]]
    Prev: [[-1001.0, -150]]
ne02:
    Next: [[-548.9, -150], [-334.7, -150], [-120.6, -150], [93.6, -150],
            ↪ [307.7, -150], [521.9, -150]]
    Prev: [[-548.9, -150], [-334.7, -150], [-120.6, -150], [93.6, -150],
            ↪ [307.7, -150], [521.9, -150]]
ne03:
    Next: [[-111.0, 150], [336.0, 150]]
    Prev: [[89.0, 150], [-363.0, 150]]
ne04:
    Next: [[1000.0, -150]]
    Prev: [[1200.0, -150]]
ne05:
    Next: [[-1218.0, -450]]
    Prev: [[-1018.0, -450]]
ne06:
    Next: [[-548.9, -450], [-334.7, -450], [-120.6, -450], [93.6, -450],
            ↪ [307.7, -450], [521.9, -450]]
    Prev: [[-548.9, -450], [-334.7, -450], [-120.6, -450], [93.6, -450],
            ↪ [307.7, -450], [521.9, -450]]

```

```

    ↗ [307.7, -450], [521.9, -450]
ne07:
  Next: [[-105.0, -750], [336.0, -750]]
  Prev: [[95.0, -750], [-363.0, -750]]
ne08:
  Next: [[1000.0, -450]]
  Prev: [[1200.0, -450]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura D.3.

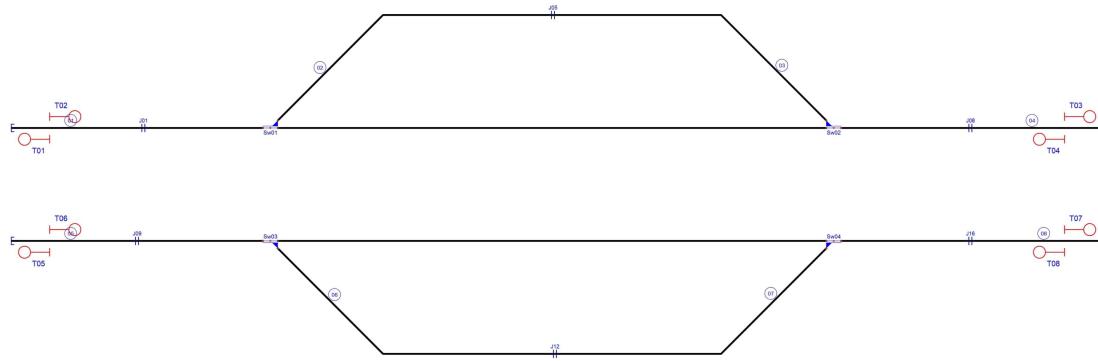


Figura D.3: Señalamiento generado por el RNA para proteger el fin de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01, T03, T05 y T07, y las señales de partida son T02, T04, T06 y T08. No existen finales de vías relativos que proteger.

La Figura D.4 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Estas señales se obtuvieron al aplicar el Algoritmo 8, tal como fue explicado en la Sección 2.2.2. Las señales generadas son todas las señales entre J09 y J20, indicadas en color rojo.

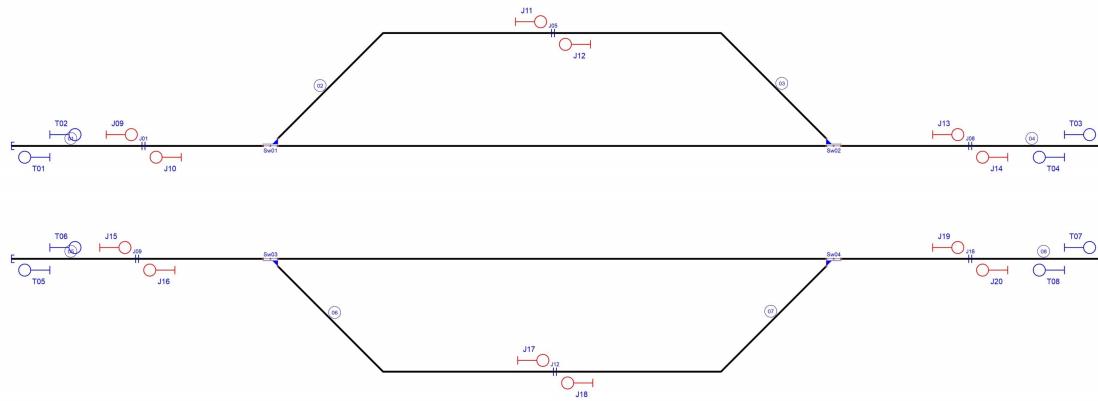


Figura D.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos. El señalamiento generado para proteger las plataformas y los cruces de vía, producto de aplicar el Algoritmo 9 y el Algoritmo 10, respectivamente, se ilustra en rojo en la Figura D.5. Al no existir plataformas o cruces de vías que proteger, ninguna señal fue generada por el RNA.

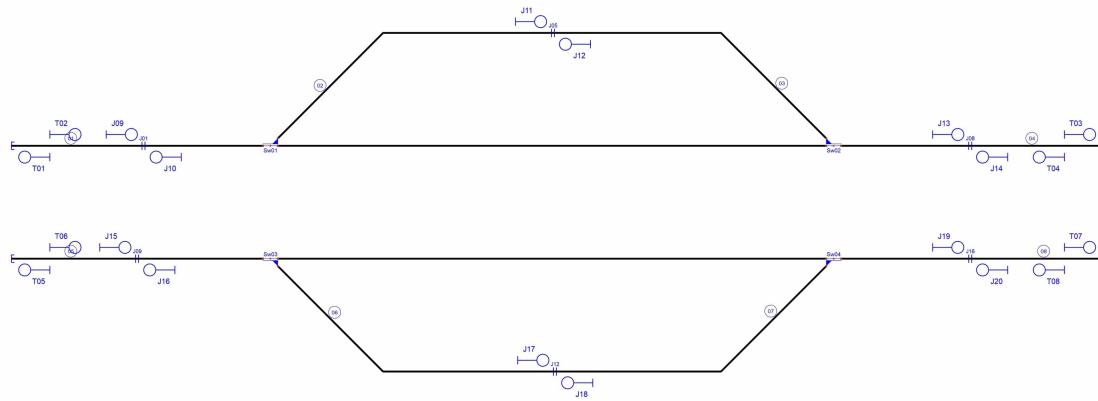


Figura D.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA genera las señales C21, S23, B26 y H24 para proteger el cambio de vías Sw02; las señales C25, S27, B22 y H28 para proteger el cambio de vías Sw02; las señales C29, S31, B34 y H32 para proteger el cambio de vías Sw03 y las señales C33, S35, B30 y H36 para proteger el cambio de vías Sw04. Las señales mencionadas se encuentran resaltadas en rojo en la Figura D.6.

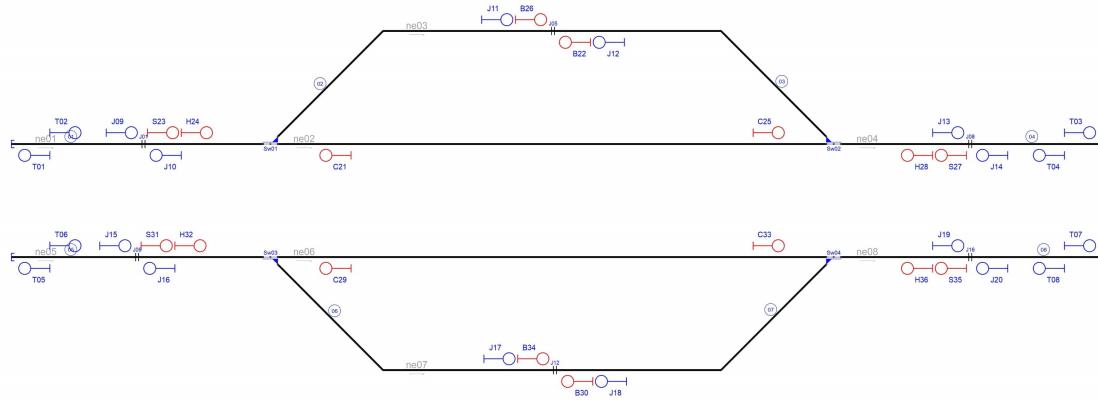


Figura D.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. En este ejemplo, el Algoritmo 15 de herencia vertical no fue aplicado, al no cumplirse las condiciones de aplicación.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: J09, J10, J11, J12, J13, J14, J15, J16, J17, J18, S23, H24, S27, H28, S31, H32, S35 y H23. Las señales J09, S23 y H24 fueron eliminadas por su cercanía con la señal T02, con la cual comparten dirección y sentido. Lo mismo ocurre entre las señales J14, S27 y H28, borradas por la señal T04; entre las señales J15, S31 y H32, borradas por la señal T06; entre las señales J20, S35 y H36, borradas por la señal T08; entre la señal J10 y la señal T01; la señal J13 y la señal T03; la señal J16 y la señal T05; y entre la señal J19 y la señal T07. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código D.5.

Código D.5: Reducción de señalamiento por prioridad de señales

```

Reducing redundant signals
removing sig10 for sig01
removing sig09 for sig02
removing sig23 for sig02
removing sig24 for sig02
removing sig13 for sig03
removing sig14 for sig04
removing sig27 for sig04
removing sig28 for sig04
removing sig16 for sig05
removing sig15 for sig06
removing sig31 for sig06
removing sig32 for sig06
removing sig19 for sig07
removing sig20 for sig08
removing sig35 for sig08
removing sig36 for sig08
  
```

```

removing sig11 for sig26
removing sig12 for sig22
removing sig17 for sig34
removing sig18 for sig30

```

El resultado de la simplificación del señalamiento se ilustra en la Figura D.7.

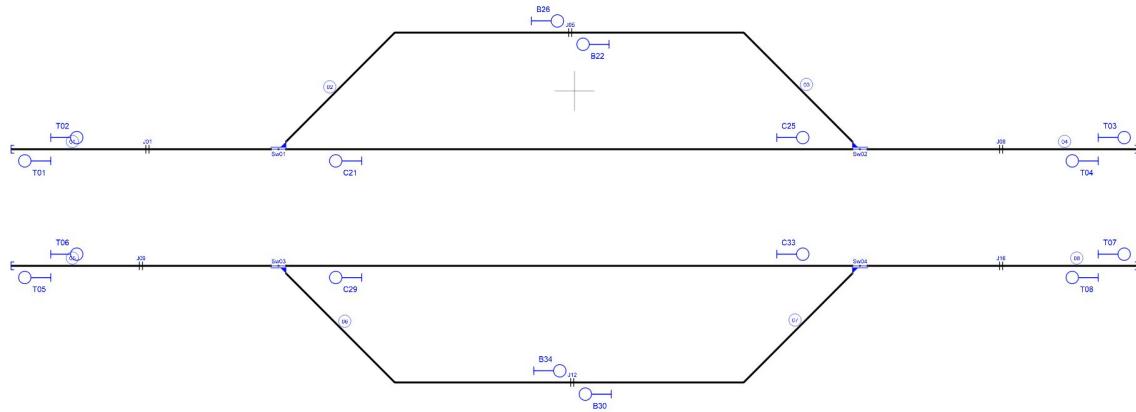


Figura D.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código D.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código D.6: Signalling.RNA

```

sig01 [T01] <<:
  From: ne01 | To: bus174_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-1351, 150] | Coordinate: 0.1453
sig02 [T02] >>:
  From: ne01 | To: ne01_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-1351, 150] | Coordinate: 0.1453
sig03 [T03] >>:
  From: ne04 | To: bus176_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [1351, 150] | Coordinate: 0.8601
sig04 [T04] <<:
  From: ne04 | To: ne04_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [1351, 150] | Coordinate: 0.8601
sig05 [T05] <<:
  From: ne05 | To: bus175_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-1351, 450] | Coordinate: 0.1453
sig06 [T06] >>:
  From: ne05 | To: ne05_right

```

```

        Type: Stop | Direction: normal | AtTrack: left
        Position: [-1351, 450] | Coordinate: 0.1453
sig07 [T07] >>:
    From: ne08 | To: bus177_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [1351, 450] | Coordinate: 0.8601
sig08 [T08] <<:
    From: ne08 | To: ne08_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [1351, 450] | Coordinate: 0.8601
sig21 [C21] <<:
    From: ne02 | To: ne02_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-548.9, 150] | Coordinate: 0.1428
sig22 [B22] <<:
    From: ne03 | To: ne03_left
    Type: Manouver | Direction: reverse | AtTrack: right
    Position: [89.0, -150] | Coordinate: 0.8014
sig25 [C25] >>:
    From: ne02 | To: ne02_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [521.9, 150] | Coordinate: 0.8571
sig26 [B26] >>:
    From: ne03 | To: ne03_right
    Type: Manouver | Direction: normal | AtTrack: left
    Position: [-111.0, -150] | Coordinate: 0.6869
sig29 [C29] <<:
    From: ne06 | To: ne06_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-548.9, 450] | Coordinate: 0.1428
sig30 [B30] <<:
    From: ne07 | To: ne07_left
    Type: Manouver | Direction: reverse | AtTrack: right
    Position: [95.0, 750] | Coordinate: 0.8048
sig33 [C33] >>:
    From: ne06 | To: ne06_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [521.9, 450] | Coordinate: 0.8571
sig34 [B34] >>:
    From: ne07 | To: ne07_right
    Type: Manouver | Direction: normal | AtTrack: left
    Position: [-105.0, 750] | Coordinate: 0.6904

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código D.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código D.7: Routes.RNA

```
route_1 [sig02 >> sig25]:  
    Path: ['ne01', 'ne02']  
    Switches: ['Sw01']  
route_2 [sig02 >> sig26]:  
    Path: ['ne01', 'ne03']  
    Switches: ['Sw01']  
route_3 [sig04 << sig21]:  
    Path: ['ne04', 'ne02']  
    Switches: ['Sw02']  
route_4 [sig04 << sig22]:  
    Path: ['ne04', 'ne03']  
    Switches: ['Sw02']  
route_5 [sig06 >> sig33]:  
    Path: ['ne05', 'ne06']  
    Switches: ['Sw03']  
route_6 [sig06 >> sig34]:  
    Path: ['ne05', 'ne07']  
    Switches: ['Sw03']  
route_7 [sig08 << sig29]:  
    Path: ['ne08', 'ne06']  
    Switches: ['Sw04']  
route_8 [sig08 << sig30]:  
    Path: ['ne08', 'ne07']  
    Switches: ['Sw04']  
route_9 [sig21 << sig01]:  
    Path: ['ne02', 'ne01']  
    Switches: ['Sw01']  
route_10 [sig22 << sig01]:  
    Path: ['ne03', 'ne01']  
    Switches: ['Sw01']  
route_11 [sig25 >> sig03]:  
    Path: ['ne02', 'ne04']  
    Switches: ['Sw02']  
route_12 [sig26 >> sig03]:  
    Path: ['ne03', 'ne04']  
    Switches: ['Sw02']  
route_13 [sig29 << sig05]:  
    Path: ['ne06', 'ne05']  
    Switches: ['Sw03']  
route_14 [sig30 << sig05]:  
    Path: ['ne07', 'ne05']  
    Switches: ['Sw03']  
route_15 [sig33 >> sig07]:  
    Path: ['ne06', 'ne08']  
    Switches: ['Sw04']  
route_16 [sig34 >> sig07]:  
    Path: ['ne07', 'ne08']  
    Switches: ['Sw04']
```

D.4. Red de grafos generada por el RNA

La información exportada en el Código D.3 (Infrastructure.RNA) Código D.4 (SafePoint.RNA), Código D.6 (Signalling.RNA) y Código D.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura D.8.

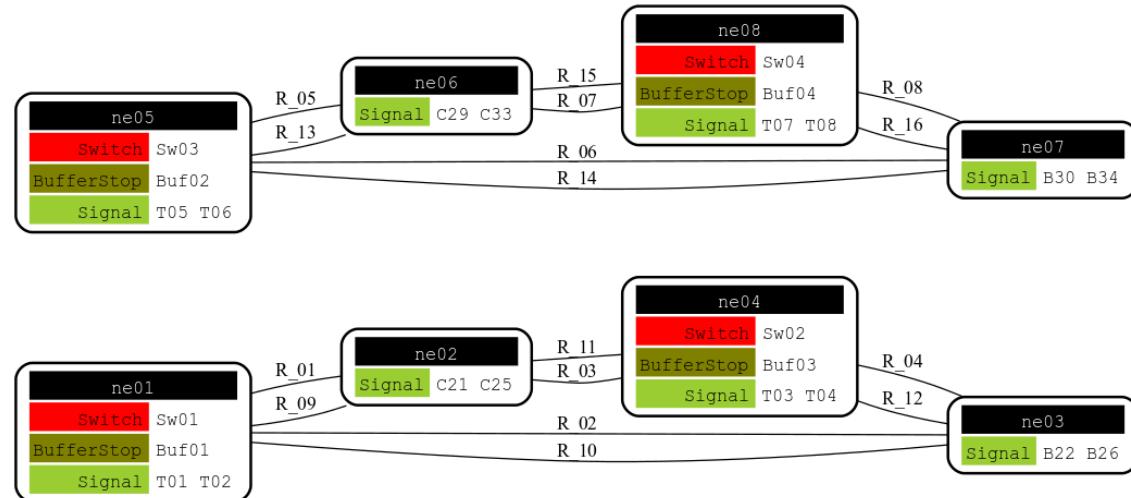


Figura D.8: Red de grafos generada por el RNA para el ejemplo 5.

Cada nodo del grafo de la Figura D.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

D.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código D.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla D.2.

En una primera inspección podemos ver que el nuevo señalamiento tiene 16 rutas, al igual que el señalamiento original que también posee 16 rutas (ver Tabla D.1). Esto se debe a que el señalamiento original ya contemplaba todas las rutas posibles y el RNA generó un nuevo señalamiento equivalente, sin eliminar rutas ni tampoco agregando rutas que no sean útiles.

D.6. Validación del sistema

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 16 rutas del señalamiento original (Tabla D.1) tienen 16 rutas equivalentes en el señalamiento generado por el RNA (Tabla D.2), tal como se puede visualizar en la Tabla D.3, generada automáticamente por el RNA.

Todas las rutas generadas por el RNA (Tabla D.2) tienen una ruta equivalente en el señalamiento original (Tabla D.1). El RNA no ha generado nuevas señales ni rutas ya que el señalamiento original ya satisfacía todos los requerimientos de seguridad necesarios. Tampoco se eliminaron

Tabla D.2: Tabla de enclavamiento del ejemplo 5 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	C ₂₅	Sw ₀₁ ^N	-	-	ne ₀₁ -ne ₀₃
R ₀₂	T ₀₂	B ₂₆	Sw ₀₁ ^R	-	-	ne ₀₁ -ne ₀₄
R ₀₃	T ₀₄	C ₂₁	Sw ₀₂ ^N	-	-	ne ₀₄ -ne ₀₃
R ₀₄	T ₀₄	B ₂₆	Sw ₀₂ ^R	-	-	ne ₀₄ -ne ₀₄
R ₀₅	T ₀₆	C ₃₃	Sw ₀₃ ^N	-	-	ne ₀₅ -ne ₀₆
R ₀₆	T ₀₆	B ₃₄	Sw ₀₃ ^R	-	-	ne ₀₅ -ne ₀₇
R ₀₇	T ₀₈	C ₂₉	Sw ₀₄ ^N	-	-	ne ₀₈ -ne ₀₆
R ₀₈	T ₀₈	B ₃₀	Sw ₀₄ ^R	-	-	ne ₀₈ -ne ₀₇
R ₀₉	C ₂₁	T ₀₁	Sw ₀₁ ^N	-	-	ne ₀₂ -ne ₀₁
R ₁₀	B ₂₂	T ₀₁	Sw ₀₁ ^R	-	-	ne ₀₃ -ne ₀₁
R ₁₁	C ₂₅	T ₀₃	Sw ₀₂ ^N	-	-	ne ₀₂ -ne ₀₄
R ₁₂	B ₂₆	T ₀₃	Sw ₀₂ ^R	-	-	ne ₀₃ -ne ₀₄
R ₁₃	C ₂₉	T ₀₅	Sw ₀₃ ^N	-	-	ne ₀₆ -ne ₀₅
R ₁₄	B ₃₀	T ₀₅	Sw ₀₃ ^R	-	-	ne ₀₇ -ne ₀₅
R ₁₅	C ₃₃	T ₀₇	Sw ₀₄ ^N	-	-	ne ₀₆ -ne ₀₈
R ₁₆	B ₃₄	T ₀₇	Sw ₀₄ ^R	-	-	ne ₀₇ -ne ₀₈

señales ni rutas, lo que demuestra que el RNA puede mejorar los sistemas de enclavamientos cuando es posible, o igualarlos cuando se ha alcanzado la cota máxima de seguridad, sin recurrir a redundancias improductivas.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

D.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura D.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código D.8.

Tabla D.3: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S ₀₁ -S ₀₆	R ₀₁	T ₀₂ -C ₂₅
R ₀₂	S ₀₅ -S ₁₃	R ₀₉	C ₂₁ -T ₀₁
R ₀₃	S ₀₁ -S ₀₃	R ₀₂	T ₀₂ -B ₂₆
R ₀₄	S ₀₄ -S ₁₃	R ₁₀	B ₂₂ -T ₀₁
R ₀₅	S ₀₂ -S ₀₄	R ₀₃	T ₀₄ -C ₂₁
R ₀₆	S ₀₆ -S ₁₅	R ₁₁	C ₂₅ -T ₀₃
R ₀₇	S ₀₂ -S ₀₅	R ₀₄	T ₀₄ -B ₂₆
R ₀₈	S ₀₃ -S ₁₅	R ₁₂	B ₂₆ -T ₀₃
R ₀₉	S ₀₇ -S ₁₁	R ₀₅	T ₀₆ -C ₃₃
R ₁₀	S ₁₀ -S ₁₄	R ₁₃	C ₂₉ -T ₀₅
R ₁₁	S ₀₇ -S ₀₉	R ₀₆	T ₀₆ -B ₃₄
R ₁₂	S ₀₈ -S ₁₄	R ₁₄	B ₃₀ -T ₀₅
R ₁₃	S ₁₂ -S ₁₀	R ₀₇	T ₀₈ -C ₂₉
R ₁₄	S ₁₁ -S ₁₆	R ₁₅	C ₃₃ -T ₀₇
R ₁₅	S ₁₂ -S ₀₈	R ₀₈	T ₀₈ -B ₃₀
R ₁₆	S ₀₉ -S ₁₆	R ₁₆	B ₃₄ -T ₀₇

Código D.8: Cantidad de elementos a implementar por el ACG

```

n_netElements:8
n_switch:4
n_doubleSwitch:0
n_borders:0
n_buffers:4
n_levelCrossings:0
n_platforms:0
n_scissorCrossings:0
n_signals:16

```

N : 44

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 62 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 5 para automatizar la importación de los archivos del ejemplo 5 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 44 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 44 en el Código D.8. El ejemplo 5 utiliza mas de 13800 sub módulos conectados automáticamente mediante mas de 27000 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla D.4. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es baja y el tiempo de síntesis e implementación es de 41 y 57 segundos, respectivamente.

Tabla D.4: Síntesis e implementación del ejemplo 5 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	2328	2307	4.38-4.34 %
FF	2344	2347	2.20-2.21 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice E

Ejemplo 6

E.1. Topología ferroviaria original

El sexto ejemplo, ilustrado en la Figura E.1, es una topología diseñada en base a una línea principal con tres niveles de ramificaciones y curvas dentro de un mismo *netElement*. Las primeras ramificaciones, utilizando los cambio de vías Sw01, Sw02 o Sw05, es una ramificación simple. La segunda ramificación, utilizando el cambio de vías Sw03, es una ramificación compleja al incluir el cambio de vías Sw01 previamente. Ademas, los *netElements* ne4, ne7, ne10 y ne41 presentan curvas que incrementan la dificultad del diseño del señalamiento. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología de múltiples ramificaciones anidadas combinadas con curvas.

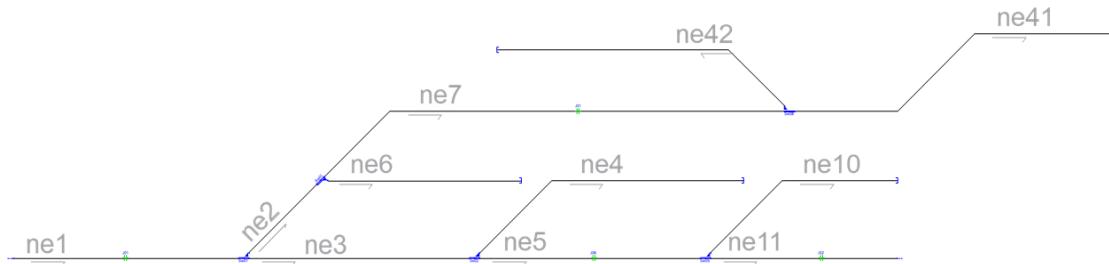


Figura E.1: Topología ferroviaria del ejemplo 6 sin señalamiento.

Para incrementar la dificultad del análisis y obtener resultados mas completos, se incluyeron finales de vías relativos y absolutos. No se incluyeron plataformas o cruces de vías.

E.2. Señalamiento original

El señalamiento original, ilustrado en la Figura E.2, incluye señales de parada próximas a los finales de vías absolutos (S11, S12, S13, S14, S16), señales de maniobras antes de converger en una vía principal (S10, S15, S20) y señales múltiples para cambios de vías divergentes (S01, S06, S07, S21), entre varias otras señales.

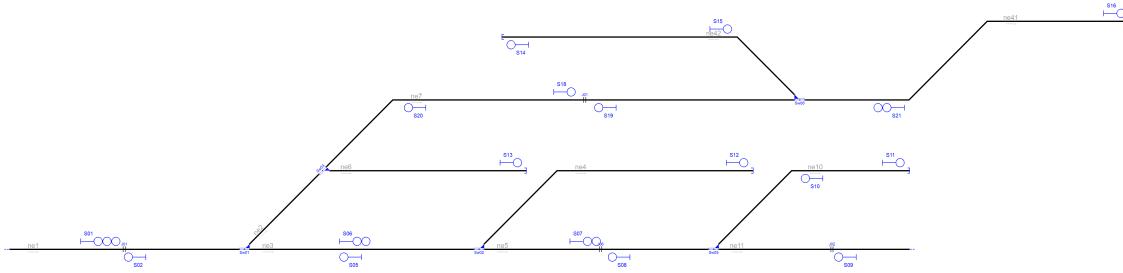


Figura E.2: Señalamiento original del ejemplo 6.

Estas señales permiten definir hasta un máximo de 16 rutas, todas ellas detalladas en la Tabla E.1. En una primera inspección, se puede comprobar que todos los elementos ferroviarios son alcanzados por al menos una de las rutas, en al menos una dirección. Además, todos los cambios de vías son utilizados, de forma simple o compuesta.

Tabla E.1: Tabla de enclavamiento original del ejemplo 6.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	S ₀₁	S ₀₆	Sw ₀₁ ^N	-	-	ne ₀₁ -ne ₀₃
R ₀₂	S ₀₁	S ₁₃	Sw ₀₁ ^R +Sw ₀₃ ^R	-	-	ne ₀₁ -ne ₀₆
R ₀₃	S ₀₁	S ₁₈	Sw ₀₁ ^R +Sw ₀₃ ^R	-	-	ne ₀₁ -ne ₀₇
R ₀₄	S ₀₆	S ₀₇	Sw ₀₂ ^N	-	-	ne ₀₃ -ne ₀₅
R ₀₅	S ₀₆	S ₁₂	Sw ₀₂ ^R	-	-	ne ₀₃ -ne ₀₄
R ₀₆	S ₂₁	S ₁₉	Sw ₀₈ ^N	-	-	ne ₄₁ -ne ₀₇
R ₀₇	S ₂₁	S ₁₄	Sw ₀₈ ^R	-	-	ne ₄₁ -ne ₄₂
R ₀₈	S ₀₅	S ₀₂	Sw ₀₁ ^N	-	-	ne ₀₃ -ne ₀₁
R ₀₉	S ₀₉	S ₀₈	Sw ₀₅ ^N	-	-	ne ₁₁ -ne ₀₅
R ₁₀	S ₀₈	S ₀₅	Sw ₀₂ ^N	-	-	ne ₀₅ -ne ₀₃
R ₁₁	S ₁₀	S ₀₈	Sw ₀₅ ^R	-	-	ne ₁₀ -ne ₀₅
R ₁₂	S ₁₅	S ₁₆	Sw ₀₈ ^R	-	-	ne ₄₂ -ne ₄₁
R ₁₃	S ₁₈	S ₁₆	Sw ₀₈ ^N	-	-	ne ₀₇ -ne ₄₁
R ₁₄	S ₁₉	S ₂₀	-	-	-	ne ₀₇
R ₁₅	S ₂₀	S ₀₂	Sw ₀₁ ^R +Sw ₀₃ ^N	-	-	ne ₀₇ -ne ₀₁
R ₁₆	S ₀₇	S ₁₁	Sw ₀₅ ^R	-	-	ne ₀₅ -ne ₁₀

Algunas rutas abarcan mas de un *netElement*, como por ejemplo la ruta R15 que comienza en la señal S20 y finaliza en la señal S02, atravesando los *netElements* ne07 y ne01, utilizando los cambios de vías Sw01 y Sw03, en posición reversa y normal respectivamente.

E.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la

conexidad de la red. El resultado obtenido se muestra en el Código E.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código E.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analyzing railML object
Analyzing graph
ne1 [-1500, 450] [-600, 450] >>
ne2 [-600, 450] [-300, 750] >>
ne3 [-600, 450] [300, 450] >>
ne4 [300, 450] [1350, 750] >>
ne5 [300, 450] [1200, 450] >>
ne6 [-300, 750] [480, 750] >>
ne7 [-300, 750] [1530, 1020] >>
ne10 [1200, 450] [1950, 750] >>
ne11 [1200, 450] [1950, 450] >>
ne41 [1530, 1020] [2790, 1320] >>
ne42 [1530, 1020] [390, 1260] <<
The network is connected
```

Por ejemplo, el *netElement* ne42 inicia en la coordenada (1530;1020) y finaliza en la coordenada (390;1260). El símbolo << indica que ne42 se encuentra definido de derecha a izquierda, ya que la componente x de la coordenada final es mayor a la de la coordenada inicial. En este caso la componente y es diferente en ambas coordenadas, ya que el *netElement* ne42, tal como se aprecia en la Figura E.1, es una curva.. Además, se puede comprobar que la lista obtenida es consistente con la Figura E.2. Por ejemplo, ne1, ne2 y ne3 comparten la coordenada (-600;450), que coincide con la coordenada del cambio de vías Sw01.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código E.2.

Código E.2: Detección de puntos críticos por parte del RNA

```
Analyzing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne1 has a RailJoint[J01] @ [-1060, 450]
ne3 has a Middle point @ [-375.0, 450]
ne3 has a Middle point @ [-150.0, 450]
ne3 has a Middle point @ [75.0, 450]
ne4 has a Curve(2 lines) @ [[600, 750]]
ne5 has a RailJoint[J06] @ [765, 450]
ne6 has a Middle point @ [-40.0, 750]
ne6 has a Middle point @ [220.0, 750]
ne7 has a RailJoint[J01] @ [705, 1020]
ne7 has a Curve(2 lines) @ [[-30, 1020]]
```

```

ne10 has a Curve(2 lines) @ [[1500, 750]]
ne11 has a RailJoint[J02] @ [1653, 450]
ne41 has a Curve(3 lines) @ [[1950, 1020], [2250, 1320]]
ne42 has a Curve(2 lines) @ [[1290, 1260]]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código E.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código E.3: Infrastructure.RNA

```

Nodes: 11|Switches: 5|Signals: 0|Detectors: 4|Ends: 7|Barriers: 0
Node ne1:
    Track = track1
    TrainDetectionElements -> tde16
        Type -> insulatedRailJoint
    Neighbours = 2 -> ['ne3', 'ne2']
    Switches -> Sw01
        ContinueCourse -> right -> ne3
        BranchCourse -> left -> ne2
Node ne2:
    Track = track7
    Neighbours = 4 -> ['ne1', 'ne3', 'ne7', 'ne6']
    Switches -> Sw03
        ContinueCourse -> left -> ne7
        BranchCourse -> right -> ne6
Node ne3:
    Track = track6
    Neighbours = 4 -> ['ne1', 'ne2', 'ne5', 'ne4']
    Switches -> Sw02
        ContinueCourse -> right -> ne5
        BranchCourse -> left -> ne4
Node ne4:
    Track = track5
    Type = BufferStop -> ['bus15']
    Neighbours = 2 -> ['ne3', 'ne5']
Node ne5:
    Track = track8
    TrainDetectionElements -> tde25
        Type -> insulatedRailJoint
    Neighbours = 4 -> ['ne3', 'ne4', 'ne10', 'ne11']
    Switches -> Sw05
        ContinueCourse -> right -> ne11
        BranchCourse -> left -> ne10
Node ne6:
    Track = track4
    Type = BufferStop -> ['bus14']
    Neighbours = 2 -> ['ne2', 'ne7']
Node ne7:
    Track = track9
    TrainDetectionElements -> tde67
        Type -> insulatedRailJoint

```

```

    Neighbours = 4 -> ['ne2', 'ne6', 'ne41', 'ne42']
Node ne10:
    Track = track3
    Type = BufferStop -> ['bus7']
    Neighbours = 2 -> ['ne5', 'ne11']
Node ne11:
    Track = track2
    TrainDetectionElements -> tde21
        Type -> insulatedRailJoint
    Neighbours = 2 -> ['ne5', 'ne10']
Node ne41:
    Track = track10
    Type = BufferStop -> ['bus62']
    Neighbours = 2 -> ['ne7', 'ne42']
    Switches -> Sw08
        ContinueCourse -> left -> ne7
        BranchCourse -> right -> ne42
Node ne42:
    Track = track11
    Type = BufferStop -> ['bus66']
    Neighbours = 2 -> ['ne7', 'ne41']

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se deseé. El resultado es exportado al archivo SafePoints.RNA (Código E.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código E.4: SafePoints.RNA

```

ne1:
    Next: [[-1160.0, 450]]
    Prev: [[-960.0, 450]]
ne3:
    Next: [[-375.0, 450], [-150.0, 450], [75.0, 450]]
    Prev: [[-375.0, 450], [-150.0, 450], [75.0, 450]]
ne4:
    Prev: [[700.0, 750]]
ne5:
    Next: [[665.0, 450]]
    Prev: [[865.0, 450]]
ne6:
    Next: [[-40.0, 750], [220.0, 750]]
    Prev: [[-40.0, 750], [220.0, 750]]
ne7:
    Next: [[605.0, 1020]]
    Prev: [[805.0, 1020], [70.0, 1020]]
ne10:
    Prev: [[1600.0, 750]]

```

```

ne11:
    Next: [[1553.0, 450]]
    Prev: [[1753.0, 450]]
ne41:
    Next: [[1850.0, 1020]]
    Prev: [[2350.0, 1320]]
ne42:
    Next: [[1190.0, 1260]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura E.3.

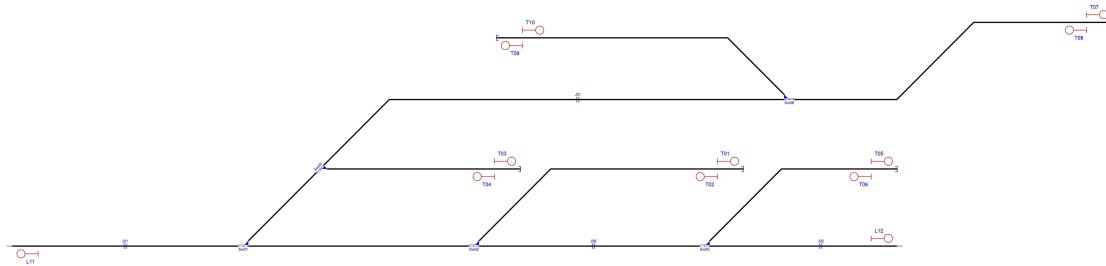


Figura E.3: Señalamiento generado por el RNA para proteger el final de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01, T03, T05, T07 y T09; y las señales de partida son T02, T04, T06, T08 y T10. A su vez, los finales de vías relativos poseen las señales de parada L11 y L12, cercanos al límite del externo del *netElement* al que pertenecen.

La Figura E.3 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Estas señales se obtuvieron al aplicar el Algoritmo 8, tal como fue explicado en la Sección 2.2.2. Las señales generadas son todas las señales comprendidas entre J13 y J20, indicadas en color rojo.

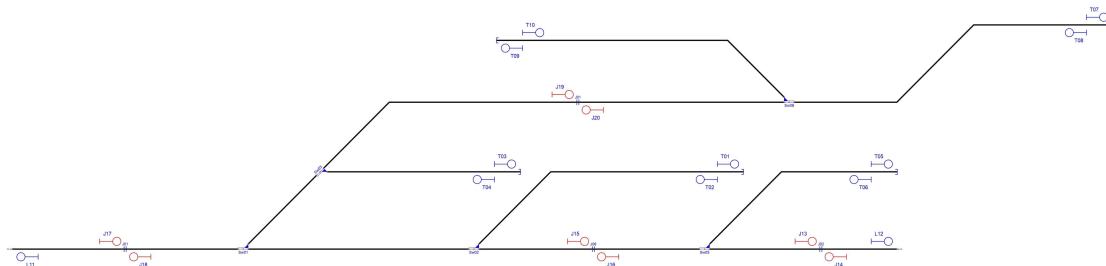


Figura E.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección

2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos. En este ejemplo, no existen plataformas o cruces de vías que proteger. Por lo tanto, el RNA no asigna nuevo señalamiento, como se visualiza en la Figura E.5.

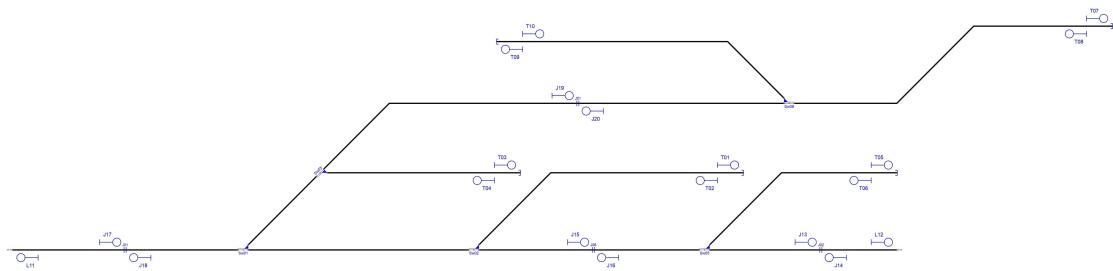


Figura E.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA genera las señales C21, S22 y H23 para proteger el cambio de vías Sw01; las señales C25, S27, B26 y H28 para proteger el cambio de vías Sw02; las señales C29, B30 Y H24 para proteger el cambio de vías Sw03; las señales C31, S33, B32 y H34 para proteger el cambio de vías Sw05 y las señales C35, S37, B36 y H38 para proteger el cambio de vías Sw08. Las señales mencionadas se encuentran resaltadas en rojo en la Figura E.6.

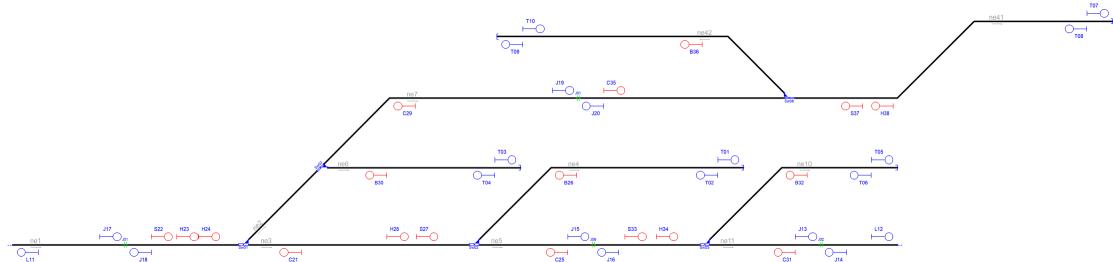


Figura E.6: Señalamiento generado por el RNA para proteger las máquinas de cambios.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en la señal B del cambio de vías Sw01, moviéndola hasta la señal C29 del cambio de vías Sw03. Además, la señal S del cambio de vías Sw03 fue movida a la señal H23 del cambio de vías Sw01.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: L12, J15, J17, H23, H24, C25, H28, B30, C31, B32, H34, C35 y H38. La señal H23 fue eliminada por su cercanía con la señal S22, con la cual comparten dirección y sentido. Lo mismo ocurre entre las señales H24 y S22; entre las señales H28 y S27; entre las señales H34 y S33; y entre las señales H38 y S37. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código E.5.

Código E.5: Reducción de señalamiento por prioridad de señales

```

Reducing redundant signals
T priority removing sig30 for sig04
T priority removing sig32 for sig06
L priority removing sig12 for sig13
J>H priority removing sig31 for sig14
J>S priority removing sig15 for sig33
J>H priority removing sig25 for sig16
J>S priority removing sig17 for sig22
J>H priority removing sig35 for sig19
Same position removing sig23 for sig22
Same position removing sig24 for sig22
Same position removing sig28 for sig27
Same position removing sig34 for sig33
Same position removing sig38 for sig37

```

El resultado de la simplificación del señalamiento se ilustra en la Figura E.7.

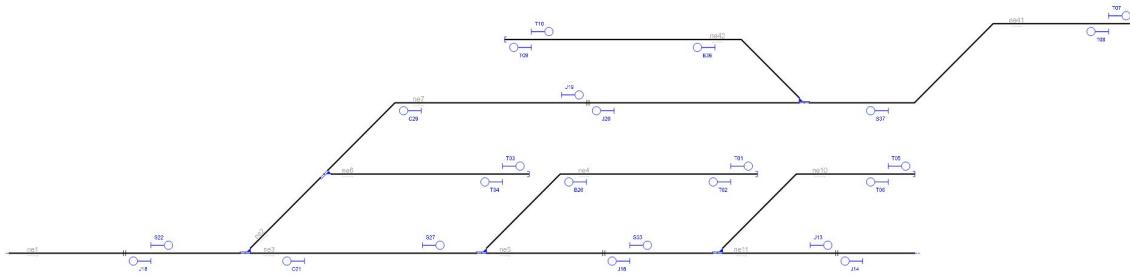


Figura E.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código 4.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código E.6: Signalling.RNA

```

sig01 [T01] >>:
    From: ne4 | To: bus15_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [1250, -750] | Coordinate: 0.9148
sig02 [T02] <<:
    From: ne4 | To: ne4_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [1250, -750] | Coordinate: 0.9148
sig03 [T03] >>:
    From: ne6 | To: bus14_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [380, -750] | Coordinate: 0.8717
sig04 [T04] <<:
    From: ne6 | To: ne6_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [380, -750] | Coordinate: 0.8717

```

```

sig05 [T05] >>:
    From: ne10 | To: bus7_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [1850, -750] | Coordinate: 0.8856
sig06 [T06] <<:
    From: ne10 | To: ne10_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [1850, -750] | Coordinate: 0.8856
sig07 [T07] >>:
    From: ne41 | To: bus62_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [2690, -1320] | Coordinate: 0.9277
sig08 [T08] <<:
    From: ne41 | To: ne41_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [2690, -1320] | Coordinate: 0.9277
sig09 [T09] <<:
    From: ne42 | To: bus66_left
    Type: Stop | Direction: normal | AtTrack: left
    Position: [490, -1260] | Coordinate: 0.3545
sig10 [T10] >>:
    From: ne42 | To: ne42_right
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [490, -1260] | Coordinate: 0.3545
sig13 [J13] >>:
    From: ne11 | To: ne11_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1553.0, -450] | Coordinate: 0.4706
sig14 [J14] <<:
    From: ne11 | To: ne11_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1753.0, -450] | Coordinate: 0.7373
sig16 [J16] <<:
    From: ne5 | To: ne5_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [865.0, -450] | Coordinate: 0.6277
sig18 [J18] <<:
    From: ne1 | To: ne1_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-960.0, -450] | Coordinate: 0.6
sig19 [J19] >>:
    From: ne7 | To: ne7_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [605.0, -1020] | Coordinate: 0.5236
sig20 [J20] <<:
    From: ne7 | To: ne7_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [805.0, -1020] | Coordinate: 0.6266
sig21 [C21] <<:
    From: ne3 | To: ne3_left
    Type: Circulation | Direction: reverse | AtTrack: right

```

```

    Position: [-375.0, -450] | Coordinate: 0.25
sig22 [S22] >>:
    From: ne1 | To: ne1_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [-960.0, -450] | Coordinate: 0.6
sig26 [B26] <<:
    From: ne4 | To: ne4_left
    Type: Manouver | Direction: reverse | AtTrack: right
    Position: [700.0, -750] | Coordinate: 0.4464
sig27 [S27] >>:
    From: ne3 | To: ne3_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [75.0, -450] | Coordinate: 0.75
sig29 [C29] <<:
    From: ne7 | To: ne7_left
    Type: Manouver | Direction: reverse | AtTrack: right
    Position: [70.0, -1020] | Coordinate: 0.2481
sig33 [S33] >>:
    From: ne5 | To: ne5_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [865.0, -450] | Coordinate: 0.6277
sig36 [B36] <<:
    From: ne42 | To: ne42_right
    Type: Manouver | Direction: normal | AtTrack: left
    Position: [1190.0, -1260] | Coordinate: 0.9193
sig37 [S37] <<:
    From: ne41 | To: ne41_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1850.0, -1020] | Coordinate: 0.9277

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código E.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código E.7: Routes.RNA

```

route_1 [sig02 << sig26]:
    Path: ['ne4']
route_2 [sig04 << sig18]:
    Path: ['ne6', 'ne2', 'ne1']
    Switches: ['Sw01', 'Sw03']
route_3 [sig06 << sig16]:
    Path: ['ne10', 'ne5']
    Switches: ['Sw05']
route_4 [sig08 << sig37]:
    Path: ['ne41']
    Switches: ['Sw08']
route_5 [sig10 >> sig07]:
    Path: ['ne42', 'ne41']

```

```

        Switches: ['Sw08']
route_6 [sig14 << sig16]:
    Path: ['ne11', 'ne5']
    Switches: ['Sw05']
route_7 [sig16 << sig21]:
    Path: ['ne5', 'ne3']
    Switches: ['Sw02', 'Sw05']
route_8 [sig19 >> sig07]:
    Path: ['ne7', 'ne41']
    Switches: ['Sw08']
route_9 [sig20 << sig29]:
    Path: ['ne7']
route_10 [sig21 << sig18]:
    Path: ['ne3', 'ne1']
    Switches: ['Sw01', 'Sw02']
route_11 [sig22 >> sig27]:
    Path: ['ne1', 'ne3']
    Switches: ['Sw01', 'Sw02']
route_12 [sig22 >> sig19]:
    Path: ['ne1', 'ne2', 'ne7']
    Switches: ['Sw01', 'Sw03']
route_13 [sig22 >> sig03]:
    Path: ['ne1', 'ne2', 'ne6']
    Switches: ['Sw01', 'Sw03']
route_14 [sig26 << sig21]:
    Path: ['ne4', 'ne3']
    Switches: ['Sw02']
route_15 [sig27 >> sig33]:
    Path: ['ne3', 'ne5']
    Switches: ['Sw02', 'Sw05']
route_16 [sig27 >> sig01]:
    Path: ['ne3', 'ne4']
    Switches: ['Sw02']
route_17 [sig29 << sig18]:
    Path: ['ne7', 'ne2', 'ne1']
    Switches: ['Sw01', 'Sw03']
route_18 [sig33 >> sig05]:
    Path: ['ne5', 'ne10']
    Switches: ['Sw05']
route_19 [sig33 >> sig13]:
    Path: ['ne5', 'ne11']
    Switches: ['Sw05']
route_20 [sig36 << sig09]:
    Path: ['ne42']
route_21 [sig37 << sig20]:
    Path: ['ne41', 'ne7']
    Switches: ['Sw08']
route_22 [sig37 << sig36]:
    Path: ['ne41', 'ne42']
    Switches: ['Sw08']

```

E.4. Red de grafos generada por el RNA

La información exportada en el Código E.3 (Infrastructure.RNA), Código E.4 (SafePoint.RNA), Código E.6 (Signalling.RNA) y Código E.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura E.8.

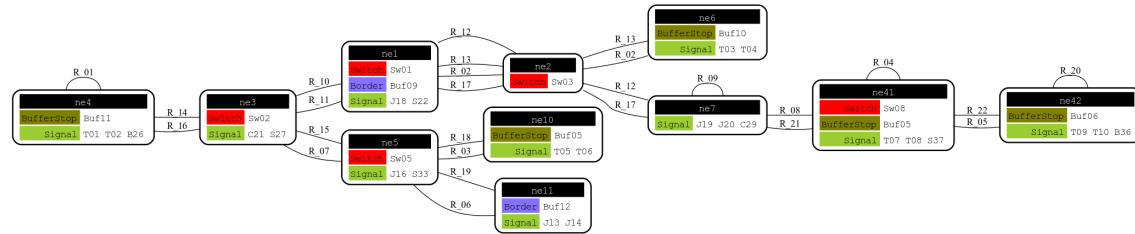


Figura E.8: Red de grafos generada por el RNA para el ejemplo 6.

Cada nodo del grafo de la Figura E.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

E.5. Señalamiento generado por el RNA

El RNA también exporta la información mostrada en el Código E.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla E.2.

Tabla E.2: Tabla de enclavamiento del ejemplo 6 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	B ₂₆	-	-	-	ne ₀₄
R ₀₂	T ₀₄	J ₁₈	Sw ₀₁ ^R +Sw ₀₃ ^R	-	-	ne ₀₆ -ne ₀₁
R ₀₃	T ₀₆	J ₁₆	Sw ₀₅ ^R	-	-	ne ₁₀ -ne ₀₅
R ₀₄	T ₀₈	S ₃₇	-	-	-	ne ₄₁
R ₀₅	T ₁₀	T ₀₇	Sw ₀₈ ^R	-	-	ne ₄₂ -ne ₄₁
R ₀₆	J ₁₄	J ₁₆	Sw ₀₅ ^N	-	-	ne ₁₁ -ne ₀₅
R ₀₇	J ₁₆	C ₂₁	Sw ₀₂ ^N	-	-	ne ₀₅ -ne ₀₃
R ₀₈	J ₁₉	T ₀₇	Sw ₀₈ ^N	-	-	ne ₀₇ -ne ₄₁
R ₀₉	J ₂₀	C ₂₉	-	-	-	ne ₀₇
R ₁₀	C ₂₁	J ₁₈	Sw ₀₁ ^N	-	-	ne ₀₃ -ne ₀₁
R ₁₁	S ₂₂	S ₂₇	Sw ₀₁ ^N	-	-	ne ₀₁ -ne ₀₃
R ₁₂	S ₂₂	J ₁₉	Sw ₀₁ ^R +Sw ₀₃ ^N	-	-	ne ₀₁ -ne ₀₇
R ₁₃	S ₂₂	T ₀₃	Sw ₀₁ ^R +Sw ₀₃ ^R	-	-	ne ₀₁ -ne ₀₆
R ₁₄	B ₂₆	C ₂₁	Sw ₀₂ ^R	-	-	ne ₀₄ -ne ₀₃
R ₁₅	S ₂₇	S ₃₃	Sw ₀₂ ^N	-	-	ne ₀₃ -ne ₀₅
R ₁₆	S ₂₇	T ₀₁	Sw ₀₂ ^R	-	-	ne ₀₃ -ne ₀₄
R ₁₇	C ₂₉	J ₁₈	Sw ₀₁ ^R +Sw ₀₃ ^N	-	-	ne ₀₇ -ne ₀₁
R ₁₈	S ₃₃	J ₁₃	Sw ₀₅ ^N	-	-	ne ₀₅ -ne ₁₁
R ₁₉	S ₃₃	T ₀₅	Sw ₀₅ ^R	-	-	ne ₀₅ -ne ₁₀
R ₂₀	B ₃₆	T ₀₉	-	-	-	ne ₄₂
R ₂₁	S ₃₇	J ₂₀	Sw ₀₈ ^N	-	-	ne ₄₁ -ne ₀₇
R ₂₂	S ₃₇	B ₃₆	Sw ₀₈ ^R	-	-	ne ₄₁ -ne ₄₂

En una primera inspección podemos ver que el nuevo señalamiento tiene 22 rutas, versus las 16 rutas del señalamiento original (ver Tabla E.1). Esto se debe a que todas las vías son consideradas de ambos sentidos por el RNA, lo cuál queda de manifiesto cuando se comprueba que todas las plataformas y cruces de vía son atravesados por dos rutas, una en cada dirección.

E.6. Validación del sistema

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 16 rutas del señalamiento original (Tabla E.1) tienen 16 rutas equivalentes en el señalamiento generado por el RNA (Tabla E.2), tal como se puede visualizar en la Tabla E.3, generada automáticamente por el RNA.

Tabla E.3: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S ₀₁ -S ₀₆	R ₁₁	S ₂₂ -S ₂₇
R ₀₂	S ₀₁ -S ₁₃	R ₁₃	S ₂₂ -T ₀₃
R ₀₃	S ₀₁ -S ₁₈	R ₁₂	S ₂₂ -J ₁₉
R ₀₄	S ₀₆ -S ₀₇	R ₁₅	S ₂₇ -S ₃₃
R ₀₅	S ₀₆ -S ₁₂	R ₁₆	S ₂₇ -T ₀₁
R ₀₆	S ₂₁ -S ₁₉	R ₂₁	S ₃₇ -J ₂₀
R ₀₇	S ₂₁ -S ₁₄	R ₂₂	S ₃₇ -B ₃₆
R ₀₈	S ₀₅ -S ₀₂	R ₁₀	C ₂₁ -J ₁₈
R ₀₉	S ₀₉ -S ₀₈	R ₀₆	J ₁₄ -J ₁₆
R ₁₀	S ₀₈ -S ₀₅	R ₀₇	J ₁₆ -C ₂₁
R ₁₁	S ₁₀ -S ₀₈	R ₀₃	T ₀₆ -J ₁₆
R ₁₂	S ₁₅ -S ₁₆	R ₀₅	T ₁₀ -T ₀₇
R ₁₃	S ₁₈ -S ₁₆	R ₀₈	J ₁₉ -T ₀₇
R ₁₄	S ₁₉ -S ₂₀	R ₀₈	J ₂₀ -C ₂₉
R ₁₅	S ₂₀ -S ₀₂	R ₁₇	C ₂₉ -J ₁₈
R ₁₆	S ₀₇ -S ₁₁	R ₁₉	S ₃₃ -T ₀₅

Las rutas R1, R2, R4, R14, R18 y R20 (Tabla E.2) generadas por el RNA que no tienen equivalencias en el señalamiento original (Tabla E.1) se deben a que el RNA creó señales extras. La ruta R1 fue creada por el RNA al añadir la señal T02 para proteger el final de vía del *netElement* ne04, creando una ruta con la señal B26. La ruta R2 fue creada por el RNA al añadir la señal T04 para proteger el final de vía del *netElement* ne06, creando una ruta con la señal J18. La ruta R4 fue creada por el RNA al añadir la señal T08 para proteger el final de vía del *netElement* ne41, creando una ruta con la señal S37.

Respecto a las otras tres rutas: la ruta R14 fue creada por el RNA al añadir la señal B26 para proteger al cambio Sw02 previo a la curva del *netElement* ne04, formando una ruta con la señal C21; la ruta R18 fue creada por el RNA al añadir la señal J13 para proteger la juntura del *netElement* ne11, formando una ruta con la señal S33; la ruta R20 fue creada por el RNA al añadir la señal T09 para proteger el final de vía del *netElement* ne42, creando una ruta con la señal B36.

Los elementos indicados (finales de vías, curvas y cambios de vías), no se encontraban completamente protegidos en el señalamiento original.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.

- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

E.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura E.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código E.8.

Código E.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:11
n_switch:5
n_doubleSwitch:0
n_borders:2
n_buffers:5
n_levelCrossings:0
n_platforms:0
n_scissorCrossings:0
n_signals:24
N : 62
```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 80 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 6 para automatizar la importación de los archivos del ejemplo 6 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 62 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 62 en el Código E.8. El ejemplo 6 utiliza mas de 19870 sub módulos conectados automáticamente mediante mas de 37910 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla E.4. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es baja y el tiempo de síntesis e implementación es de 49 y 1 minuto con 10 segundos, respectivamente.

Tabla E.4: Síntesis e implementación del ejemplo 6 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	3388	3348	6.37-6.29 %
FF	3793	3796	3.56-3.57 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice F

Ejemplo 7

F.1. Topología ferroviaria original

El primer ejemplo, ilustrado en la Figura F.1, es una topología diseñada en base a múltiples niveles de ramificaciones. La primera ramificación, utilizando los cambios de vías Sw18 y Sw19, es una ramificación doble que lleva a una vía secundaria donde las formaciones pueden maniobrar para volver a la red. La segunda ramificación, utilizando los cambios de vías Sw18 y Sw14, es una ramificación compleja al incluir el cambio de vías Sw14 a continuación del Sw18. Ademas, se incluyeron curvas en los *netElements* ne41 y ne32. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología de múltiples ramificaciones que llevan a vías sin salida.

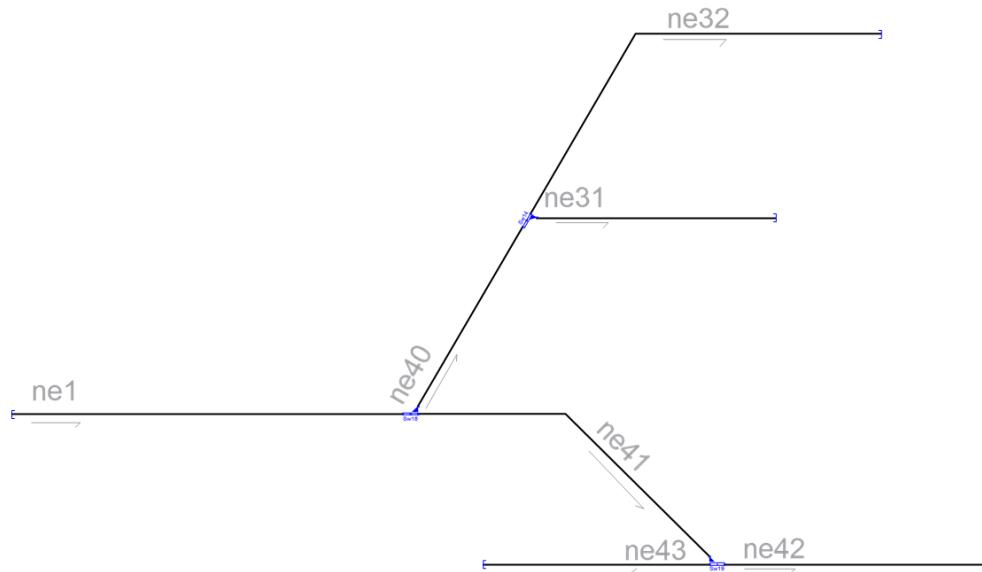


Figura F.1: Topología ferroviaria del ejemplo 7 sin señalamiento.

Para incrementar la dificultad del análisis y obtener resultados mas completos, todos los finales de vías son absolutos. No se incluyeron plataformas ni cruces de vías. La distribución de los cambios de vías se diseñó para abarcar la mayor cantidad de casos posibles.

F.2. Señalamiento original

El señalamiento original, ilustrado en la Figura F.2, incluye señales de parada próximas a los finales de vías absolutos (S06, S07, S08, S09, S10), señales de maniobras antes de converger en una vía principal (S02, S04, S05) y señales múltiples para cambios de vías divergentes (S01, S03).

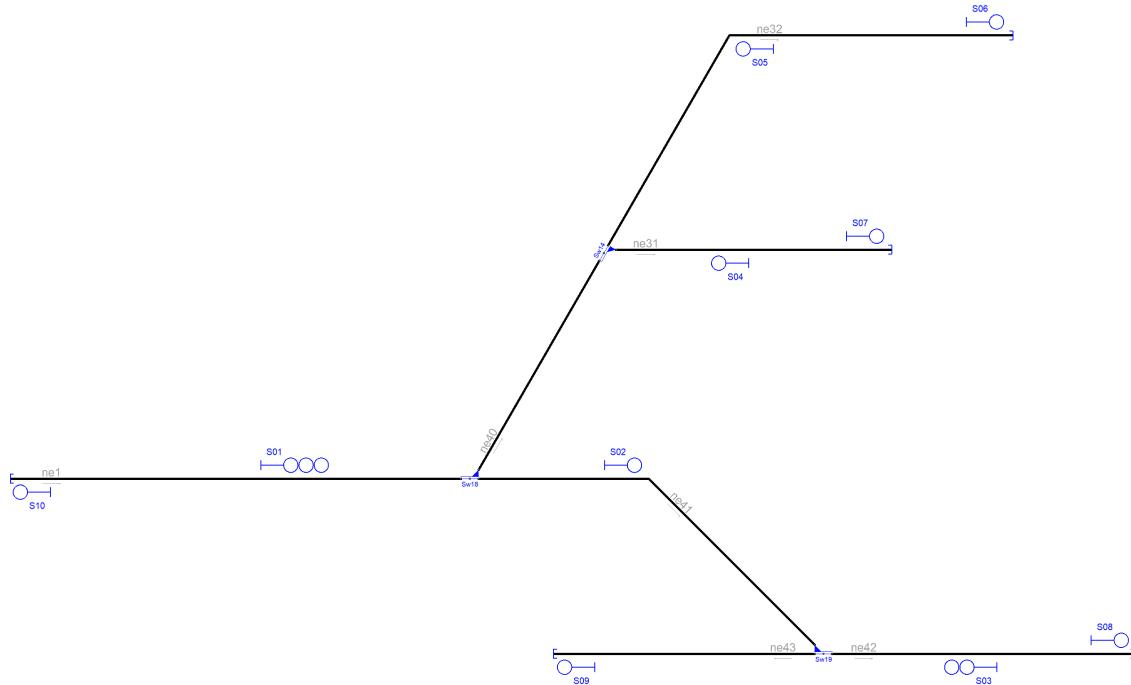


Figura F.2: Señalamiento original del ejemplo 7.

Estas señales permiten definir hasta un máximo de 8 rutas, todas ellas detalladas en la Tabla F.1. En una primera inspección, se puede comprobar que todos los elementos ferroviarios son alcanzados por al menos una de las rutas, en al menos una dirección. Además, todos los cambios de vías son utilizados, de forma simple o compuesta.

Tabla F.1: Tabla de enclavamiento original del ejemplo 7.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R01	S01	S02	Sw ^N ₁₈	-	-	ne01-ne41
R02	S01	S06	Sw ^N ₁₄ +Sw ^R ₁₈	-	-	ne01-ne32
R03	S01	S07	Sw ^R ₁₄ +Sw ^N ₁₈	-	-	ne01-ne31
R04	S03	S09	Sw ^N ₁₉	-	-	ne42-ne43
R05	S03	S10	Sw ^N ₁₈ +Sw ^R ₁₉	-	-	ne42-ne01
R06	S02	S08	Sw ^R ₁₉	-	-	ne41-ne42
R07	S04	S10	Sw ^R ₁₄ +Sw ^R ₁₈	-	-	ne31-ne01
R08	S05	S10	Sw ^N ₁₄ +Sw ^R ₁₈	-	-	ne32-ne01

Todas las rutas abarcan mas de un *netElement*, como por ejemplo la ruta R08 que comienza

en la señal S05 y finaliza en la señal S10, atravesando los *netElements* ne32 y ne01, utilizando los cambios de vías Sw14 y Sw18, en posición normal y reversa respectivamente.

F.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código F.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código F.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analyzing railML object
Analyzing graph
ne1 [-770, -30] [260, -30] >>
ne31 [554, 480] [1190, 480] >>
ne32 [554, 480] [1460, 957] >>
ne40 [260, -30] [554, 480] >>
ne41 [260, -30] [1040, -420] >>
ne42 [1040, -420] [1730, -420] >>
ne43 [1040, -420] [440, -420] <<
The network is connected
```

Por ejemplo, el *netElement* ne42 inicia en la coordenada (1040;-420) y finaliza en la coordenada (1730;-420). El símbolo >> indica que ne1 se encuentra definido de izquierda a derecha, ya que la componente x de la coordenada final es mayor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura F.2. Por ejemplo, ne1, ne40 y ne41 comparten la coordenada (260;-30), que coincide con la coordenada del cambio de vías Sw18.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los *netElements* que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código F.2.

Código F.2: Detección de puntos críticos por parte del RNA

```
Analyzing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne1 has a Middle point @ [-564.0, -30]
ne1 has a Middle point @ [-358.0, -30]
ne1 has a Middle point @ [-152.0, -30]
ne1 has a Middle point @ [54.0, -30]
ne31 has a Middle point @ [766.0, 480]
ne31 has a Middle point @ [978.0, 480]
ne32 has a Curve(2 lines) @ [[830, 957]]
```

```

ne41 has a Curve(2 lines) @ [[650, -30]]
ne42 has a Middle point @ [1270.0, -420]
ne42 has a Middle point @ [1500.0, -420]
ne43 has a Middle point @ [640.0, -420]
ne43 has a Middle point @ [840.0, -420]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código ??) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código F.3: Infrastructure.RNA

```

Nodes: 7|Switches: 3|Signals: 0|Detectors: 0|Ends: 5|Barriers: 0
Node ne1:
    Track = track1
    Type = BufferStop -> ['bus1']
    Neighbours = 2 -> ['ne41', 'ne40']
    Switches -> Sw18
        ContinueCourse -> right -> ne41
        BranchCourse -> left -> ne40
Node ne31:
    Track = track2
    Type = BufferStop -> ['bus4']
    Neighbours = 2 -> ['ne40', 'ne32']
Node ne32:
    Track = track4
    Type = BufferStop -> ['bus35']
    Neighbours = 2 -> ['ne31', 'ne40']
Node ne40:
    Track = track3
    Neighbours = 4 -> ['ne1', 'ne31', 'ne32', 'ne41']
    Switches -> Sw14
        ContinueCourse -> left -> ne32
        BranchCourse -> right -> ne31
Node ne41:
    Track = track5
    Neighbours = 4 -> ['ne1', 'ne40', 'ne42', 'ne43']
Node ne42:
    Track = track6
    Type = BufferStop -> ['bus48']
    Neighbours = 2 -> ['ne41', 'ne43']
    Switches -> Sw19
        ContinueCourse -> left -> ne43
        BranchCourse -> right -> ne41
Node ne43:
    Track = track7
    Type = BufferStop -> ['bus47']
    Neighbours = 2 -> ['ne41', 'ne42']

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación

que se desee. El resultado es exportado al archivo SafePoints.RNA (Código F.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código F.4: SafePoints.RNA

```

ne1:
    Next: [[-564.0, -30], [-358.0, -30], [-152.0, -30], [54.0, -30]]
    Prev: [[-564.0, -30], [-358.0, -30], [-152.0, -30], [54.0, -30]]
ne31:
    Next: [[766.0, 480], [978.0, 480]]
    Prev: [[766.0, 480], [978.0, 480]]
ne32:
    Prev: [[930.0, 957]]
ne41:
    Next: [[550.0, -30]]
ne42:
    Next: [[1270.0, -420], [1500.0, -420]]
    Prev: [[1270.0, -420], [1500.0, -420]]
ne43:
    Next: [[640.0, -420], [840.0, -420]]
    Prev: [[640.0, -420], [840.0, -420]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura F.3.

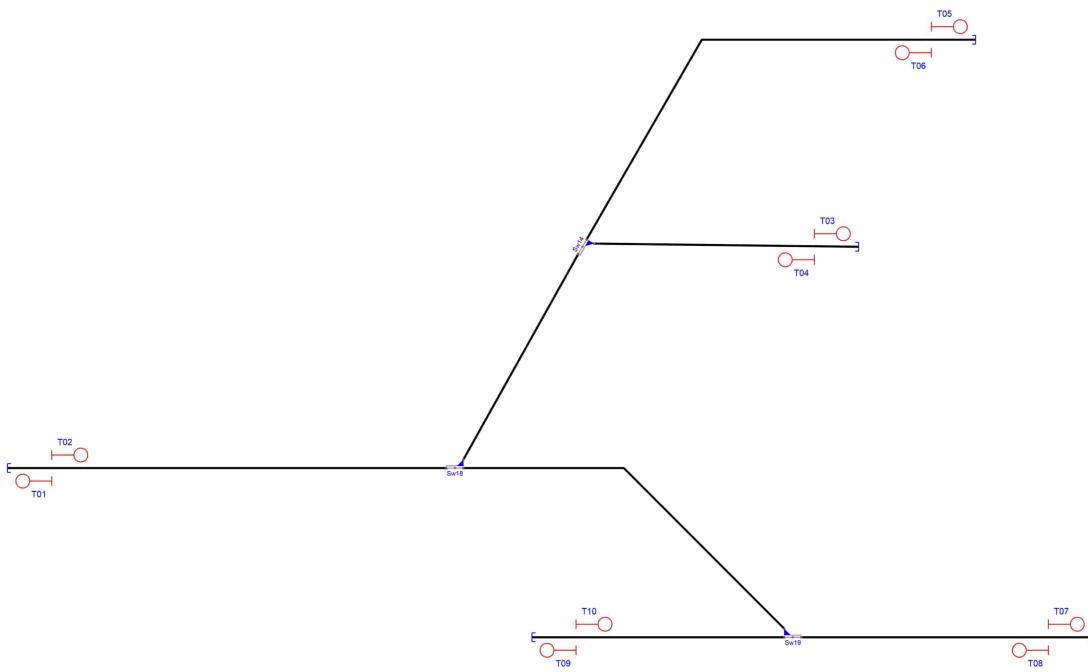


Figura F.3: Señalamiento generado por el RNA para proteger el fin de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01, T03, T05, T07 y T09; y las señales de partida son T02, T04, T06, T08 y T10. No existen finales de vías relativos que proteger, por lo que el RNA asigna señalamiento para ese fin. Al tampoco existir junturas que proteger, la Figura F.4 no presenta cambios.

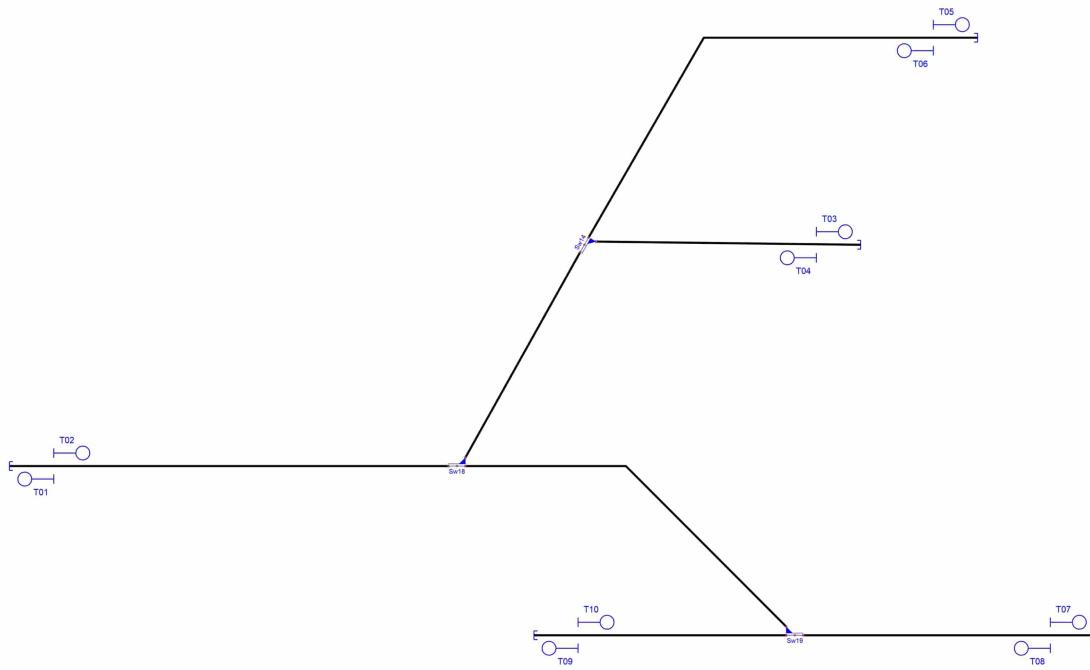


Figura F.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos. Al no existir plataformas o cruces de vías el señalamiento que se ilustra en la Figura F.5 no presenta cambios.

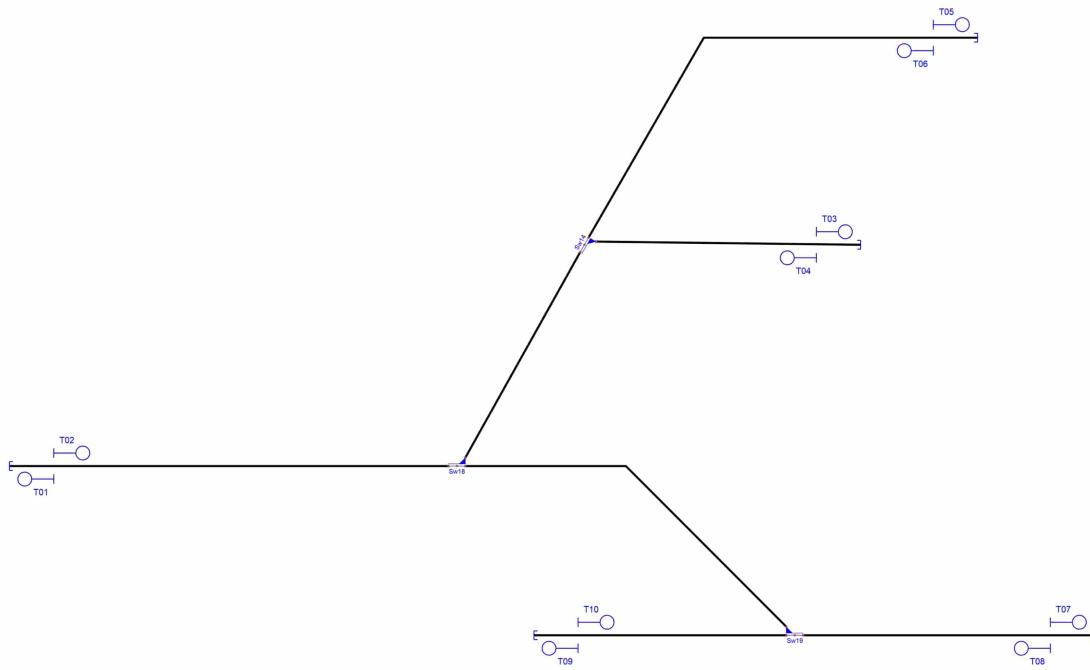


Figura F.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA genera las señales S14, C13 y H15 para proteger el cambio de vías Sw18; las señales C11, B12 y H16 para proteger el cambio de vías Sw14 y las señales S19, C17, B18 y H20 para proteger el cambio de vías Sw19. Las señales mencionadas se encuentran resaltadas en rojo en la Figura F.6.

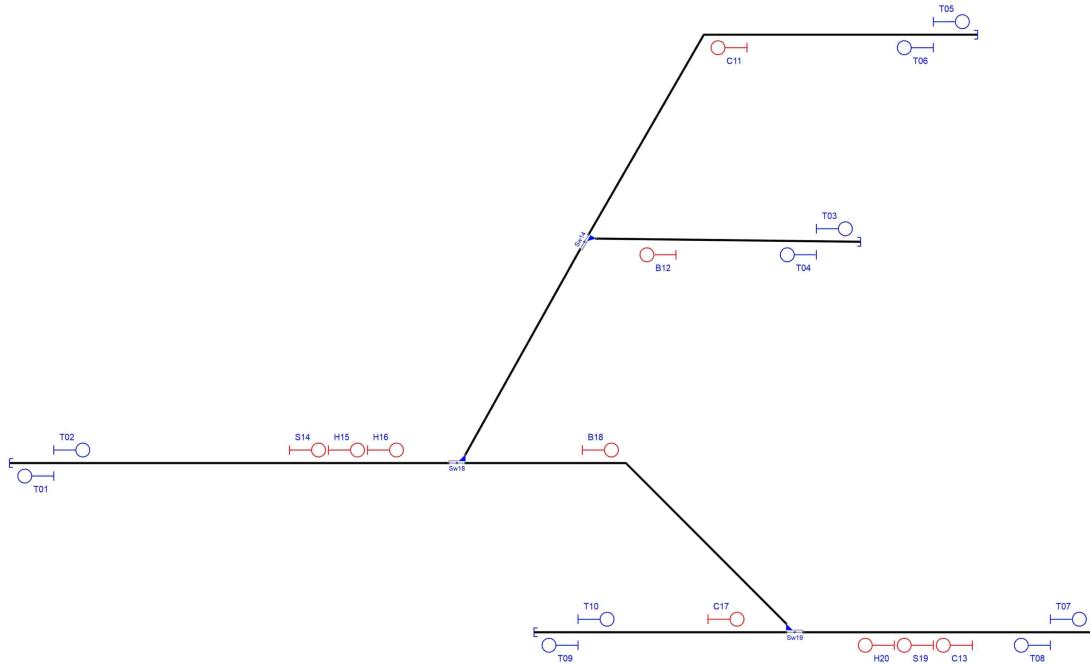


Figura F.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías Sw18 y Sw19, desplazando las señales hasta convertirlas en las señales B12 y B18 respectivamente. Análogamente, las señales C y S del *netElement* se convirtieron en las señales H15 y C17 respectivamente.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: C11, B12, H15, H16, C17, C13, S19 y H20. Las señales H15 y H16 fueron eliminadas por su cercanía con la señal S14, con la cual comparten dirección y sentido. Lo mismo ocurre entre las señales H20 y C13. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contendedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código F.5.

Código F.5: Reducción de señalamiento por prioridad de señales

```

Reducing redundant signals
T priority removing sig12 for sig04
T priority removing sig11 for sig06
T priority removing sig13 for sig08
T priority removing sig19 for sig08
  
```

T priority removing sig17 for sig10
 Same position removing sig15 for sig14
 Same position removing sig16 for sig14
 Same position removing sig20 for sig13

El resultado de la simplificación del señalamiento se ilustra en la Figura F.7.

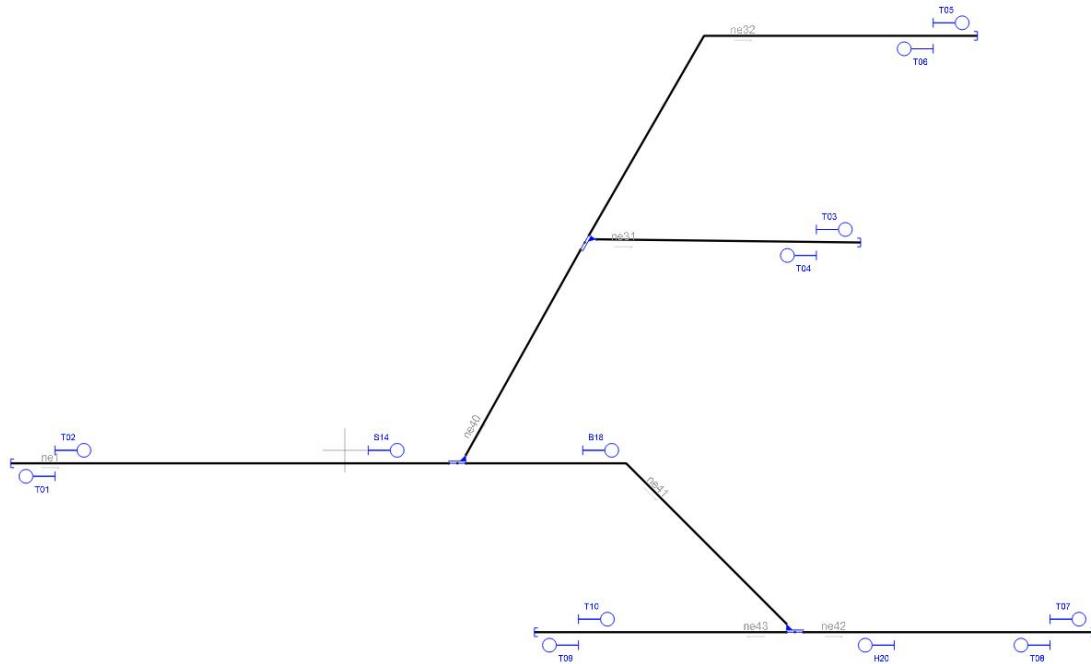


Figura F.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código F.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código F.6: Signalling.RNA

```

sig01 [T01] <<:
  From: ne1 | To: bus1_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [-670, 30] | Coordinate: 0.0970
sig02 [T02] >>:
  From: ne1 | To: ne1_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [-670, 30] | Coordinate: 0.0970
sig03 [T03] >>:
  From: ne31 | To: bus4_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [1090, -480] | Coordinate: 0.8427
sig04 [T04] <<:
  From: ne31 | To: ne31_left

```

```

    Type: Stop | Direction: reverse | AtTrack: right
    Position: [1090, -480] | Coordinate: 0.8427
sig05 [T05] >>:
    From: ne32 | To: bus35_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [1360, -957] | Coordinate: 0.9153
sig06 [T06] <<:
    From: ne32 | To: ne32_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [1360, -957] | Coordinate: 0.9153
sig07 [T07] >>:
    From: ne42 | To: bus48_right
    Type: Stop | Direction: normal | AtTrack: left
    Position: [1630, 420] | Coordinate: 0.8550
sig08 [T08] <<:
    From: ne42 | To: ne42_left
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [1630, 420] | Coordinate: 0.8550
sig09 [T09] <<:
    From: ne43 | To: bus47_left
    Type: Stop | Direction: normal | AtTrack: left
    Position: [540, 420] | Coordinate: 0.1666
sig10 [T10] >>:
    From: ne43 | To: ne43_right
    Type: Stop | Direction: reverse | AtTrack: right
    Position: [540, 420] | Coordinate: 0.1666
sig14 [S14] >>:
    From: ne1 | To: ne1_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [54.0, 30] | Coordinate: 0.8
sig18 [B18] >>:
    From: ne41 | To: ne41_right
    Type: Manouver | Direction: normal | AtTrack: left
    Position: [550.0, 30] | Coordinate: 0.8937
sig20 [H20] <<:
    From: ne42 | To: ne42_left
    Type: Manouver | Direction: reverse | AtTrack: right
    Position: [1270.0, 420] | Coordinate: 0.3333

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código F.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código F.7: Routes.RNA

```

route_1 [sig02 >> sig14]:
    Path: ['ne1']
    Switches: ['Sw18']
route_2 [sig04 << sig01]:

```

```

Path: ['ne31', 'ne40', 'ne1']
Switches: ['Sw14', 'Sw18']
route_3 [sig06 << sig01]:
    Path: ['ne32', 'ne40', 'ne1']
    Switches: ['Sw14', 'Sw18']
route_4 [sig08 << sig20]:
    Path: ['ne42']
    Switches: ['Sw19']
route_5 [sig10 >> sig07]:
    Path: ['ne43', 'ne42']
    Switches: ['Sw19']
route_6 [sig14 >> sig18]:
    Path: ['ne1', 'ne41']
    Switches: ['Sw18']
route_7 [sig14 >> sig03]:
    Path: ['ne1', 'ne40', 'ne31']
    Switches: ['Sw14', 'Sw18']
route_8 [sig14 >> sig05]:
    Path: ['ne1', 'ne40', 'ne32']
    Switches: ['Sw14', 'Sw18']
route_9 [sig18 >> sig07]:
    Path: ['ne41', 'ne42']
    Switches: ['Sw19']
route_10 [sig20 << sig01]:
    Path: ['ne42', 'ne41', 'ne1']
    Switches: ['Sw18', 'Sw19']
route_11 [sig20 << sig09]:
    Path: ['ne42', 'ne43']
    Switches: ['Sw19']

```

F.4. Red de grafos generada por el RNA

La información exportada en el Código ?? (Infrastructure.RNA) Código F.4 (SafePoint.RNA), Código F.6 (Signalling.RNA) y Código F.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura F.8.

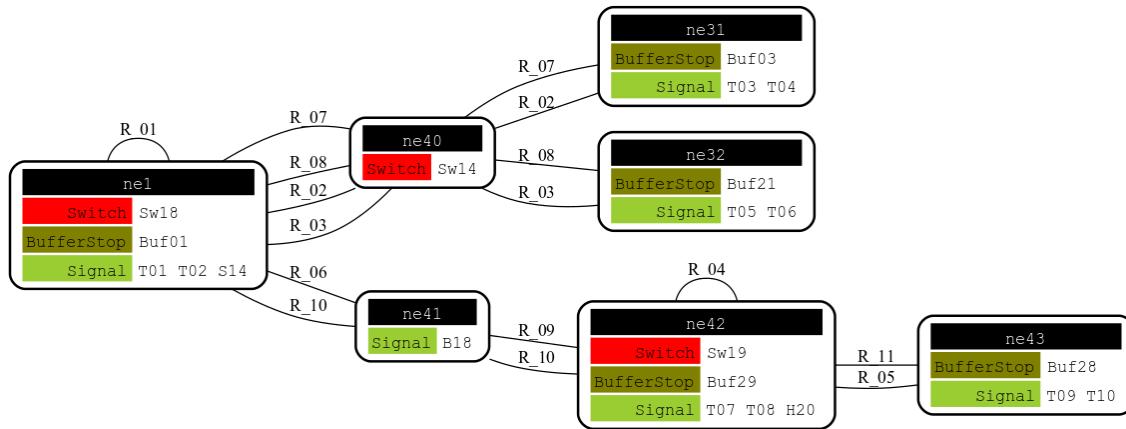


Figura F.8: Red de grafos generada por el RNA para el ejemplo 7.

Cada nodo del grafo de la Figura F.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

F.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código F.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla F.2.

Tabla F.2: Tabla de enclavamiento del ejemplo 7 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	S ₁₄	-	-	-	ne01
R ₀₂	T ₀₄	T ₀₁	Sw ₁₄ ^R +Sw ₁₈ ^R	-	-	ne31-ne01
R ₀₃	T ₀₆	T ₀₁	Sw ₁₄ ^N +Sw ₁₈ ^R	-	-	ne32-ne01
R ₀₄	T ₀₈	H ₂₀	-	-	-	ne42
R ₀₅	T ₁₀	T ₀₇	Sw ₁₉ ^N	-	-	ne43-ne42
R ₀₆	S ₁₄	B ₁₈	Sw ₁₈ ^N	-	-	ne01-ne41
R ₀₇	S ₁₄	T ₀₅	Sw ₁₈ ^R +Sw ₁₄ ^N	-	-	ne01-ne32
R ₀₈	S ₁₄	T ₀₃	Sw ₁₈ ^R +Sw ₁₄ ^R	-	-	ne01-ne31
R ₀₉	B ₁₈	T ₀₇	Sw ₁₉ ^R	-	-	ne41-ne42
R ₁₀	H ₂₀	T ₀₉	Sw ₁₉ ^N	-	-	ne42-ne43
R ₁₁	H ₂₀	T ₀₁	Sw ₁₉ ^R +Sw ₁₈ ^N	-	-	ne42-ne01

En una primera inspección podemos ver que el nuevo señalamiento tiene 11 rutas, versus las 8 rutas del señalamiento original (ver Tabla F.1). Esto se debe a que el RNA añade protecciones extras para los finales de vía absolutos, ausentes en el señalamiento original.

F.6. Validación del sistema

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 8 rutas del señalamiento original (Tabla F.1) tienen 8 rutas equivalentes en el señalamiento generado por el RNA (Tabla F.2), tal como se puede visualizar en la Tabla F.3, generada automáticamente por el RNA.

Tabla F.3: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S ₀₁ -S ₀₂	R ₀₆	S ₁₄ -B ₁₈
R ₀₂	S ₀₁ -S ₀₆	R ₀₇	S ₁₄ -T ₀₅
R ₀₃	S ₀₁ -S ₀₇	R ₀₈	S ₁₄ -T ₀₃
R ₀₄	S ₀₃ -S ₀₉	R ₀₉	B ₁₈ -T ₀₇
R ₀₅	S ₀₃ -S ₁₀	R ₁₁	H ₂₀ -T ₀₉
R ₀₆	S ₀₂ -S ₀₈	R ₁₀	H ₂₀ -T ₀₁
R ₀₇	S ₀₄ -S ₁₀	R ₀₂	T ₀₄ -T ₀₁
R ₀₈	S ₀₅ -S ₁₀	R ₀₃	T ₀₆ -T ₀₁

Las rutas R1, R4 y R5 (Tabla F.2) generadas por el RNA que no tienen equivalencias en el señalamiento original (Tabla F.1) se deben a que el RNA creó señales extras. Las señales T02, T08 y T10 fueron creadas por el RNA para proteger los finales de vía absolutos. La Ruta R1 fue establecida por el RNA al crear la señal T02 para proteger el final de vía del *netElement* ne01, creando una ruta con la señal S14. La Ruta R4 fue establecida por el RNA al crear la señal T08 para proteger el final de vía del *netElement* ne42, creando una ruta con la señal S14. La ruta R4 añade una parada intermedia entre las rutas R4 y R5 del señalamiento original, previo al cambio de vías Sw19. La Ruta R5 fue establecida por el RNA al crear la señal T10 para proteger el final de vía del *netElement* ne43, creando una ruta con la señal T07. Estos elementos ferroviarios no se encontraban protegidos en el señalamiento original, además que particionar las rutas R4 y R5 originales en R4+R10 y R4+R11 respectivamente incrementa la movilidad de la red.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

F.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura F.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código F.8.

Código F.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:7
n_switch:3
n_doubleSwitch:0
n_borders:0
n_buffers:5
n_levelCrossings:0
n_platforms:0
n_scissorCrossings:0
n_signals:13
N : 34
```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 52 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 7 para automatizar la importación de los archivos del ejemplo 7 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 34 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 34 en el Código F.8. El ejemplo 7 utiliza mas de 10800 sub módulos conectados automáticamente mediante mas de 21150 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla F.4. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es baja y el tiempo de síntesis e implementación es de 35 y 54 segundos, respectivamente.

Tabla F.4: Síntesis e implementación del ejemplo 7 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	1824	1798	3.43-3.38 %
FF	2008	2011	1.89 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice G

Ejemplo 8

G.1. Topología ferroviaria original

El primer ejemplo, ilustrado en la Figura G.1, es una topología diseñada en base a dos líneas principales conectadas por un cambio de vías y seis cruces de vías. Los cambios de vías Sw11 y Sw12 permiten el intercambio de formaciones entre ambas vías principales. El objetivo de este ejemplo fue comprobar el funcionamiento del RNA con una topología de doble vía conectadas por dos cambios de vías, similares a las que se pueden encontrar en estaciones reales.

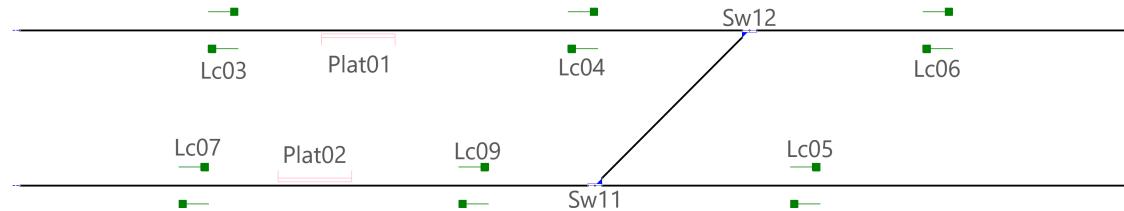


Figura G.1: Topología ferroviaria del ejemplo 8 sin señalamiento.

Para incrementar la dificultad del análisis y obtener resultados mas completos, todos los finales de vías son relativos y se añadieron las plataformas (Platform01, Platform02) y cruces de vías (Lc03, Lc04, Lc05, Lc06, Lc07, LC09). La infraestructura se distribuyó de forma tal de tener que en algunos casos el espacio entre ambos fuese suficiente (LevelCrossing04 y Platform01) y en otros no (LevelCrossing07 y Platform02).

G.2. Señalamiento original

El señalamiento original, ilustrado en la Figura G.2, incluye señales de parada próximas a los finales de vías relativos (S01, S02, S03, S04), señales de partida en las plataformas (S10, S16), señales de protección antes de cada paso a nivel (S06, S07, S08, S10, S11, S12, S13, S14, S15, S16), entre varias otras señales.

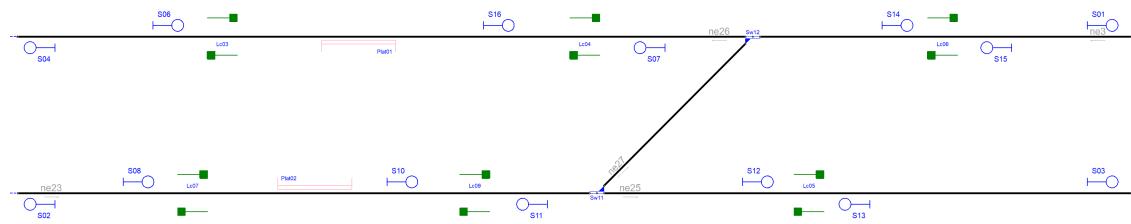


Figura G.2: Señalamiento original del ejemplo 8.

Estas señales permiten definir hasta un máximo de 12 rutas, todas ellas detalladas en la Tabla G.1. En una primera inspección, se puede comprobar que todos los elementos ferroviarios son alcanzados por al menos una de las rutas, en al menos una dirección. Además, todos los cambios de vías son utilizados, de forma simple o compuesta.

Tabla G.1: Tabla de enclavamiento original del ejemplo 8.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	S ₀₆	S ₁₆	-	Plat ₀₁	Lc ₀₃	ne ₂₆
R ₀₂	S ₀₇	S ₀₅	-	Plat ₀₁	Lc ₀₄	ne ₂₆
R ₀₃	S ₀₈	S ₁₀	-	Plat ₀₂	Lc ₀₇	ne ₂₃
R ₀₄	S ₁₀	S ₁₂	Sw ₁₁ ^N	-	Lc ₀₉	ne ₂₃ -ne ₂₅
R ₀₅	S ₁₀	S ₁₄	Sw ₁₁ ^R +Sw ₁₂ ^R	-	Lc ₀₉	ne ₂₃ -ne ₀₃
R ₀₆	S ₁₁	S ₀₉	-	Plat ₀₂	Lc ₀₉	ne ₂₃
R ₀₇	S ₁₂	S ₀₃	-	-	Lc ₀₅	ne ₂₅
R ₀₈	S ₁₃	S ₁₁	-	-	Lc ₀₅	ne ₂₅ -ne ₂₃
R ₀₉	S ₁₄	S ₀₁	-	-	Lc ₀₆	ne ₀₃
R ₁₀	S ₁₅	S ₀₇	Sw ₁₂ ^N	-	Lc ₀₆	ne ₀₃ -ne ₂₆
R ₁₁	S ₁₅	S ₁₁	Sw ₁₁ ^R +Sw ₁₂ ^N	-	Lc ₀₆	ne ₀₃ -ne ₂₃
R ₁₂	S ₁₆	S ₁₄	-	-	Lc ₀₄	ne ₂₆ -ne ₀₃

Algunas rutas abarcan mas de un *netElement*, como por ejemplo la ruta R11 que comienza en la señal S15 y finaliza en la señal S11, atravesando los *netElements* ne03 y ne23, utilizando los cambios de vías Sw11 y Sw12, en posición reversa y normal respectivamente.

G.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código G.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código G.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
```

```

Creating railML object
Analyzing railML object
Analyzing graph
ne3 [2040, 120] [1020, 120] <<
ne23 [-960, -300] [600, -300] >>
ne25 [600, -300] [2040, -300] >>
ne26 [1020, 120] [-960, 120] <<
ne27 [600, -300] [1020, 120] >>
The network is connected

```

Por ejemplo, el *netElement* ne26 inicia en la coordenada (1020;120) y finaliza en la coordenada (-960;120). El símbolo << indica que ne1 se encuentra definido de derecha a izquierda, ya que la componente x de la coordenada final es menor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura G.2. Por ejemplo, ne03, ne26 y ne27 comparten la coordenada (1020;120), que coincide con la coordenada del cambio de vías Sw12.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código G.2.

Código G.2: Detección de puntos críticos por parte del RNA

```

Analyzing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne3 has a LevelCrossing[Lc06] @ [1536, -120]
ne23 has a Platform[Plat02] @ [-160, 300]
ne23 has a LevelCrossing[Lc07] @ [-494, 300]
ne23 has a LevelCrossing[Lc09] @ [266, 300]
ne25 has a LevelCrossing[Lc05] @ [1166, 300]
ne26 has a Platform[Plat01] @ [-41, -120]
ne26 has a LevelCrossing[Lc03] @ [-404, -120]
ne26 has a LevelCrossing[Lc04] @ [572, -120]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código G.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código G.3: Infrastructure.RNA

```

Nodes: 5|Switches: 2|Signals: 0|Detectors: 0|Ends: 4|Barriers: 6
Node ne3:
    Track = track2
    Neighbours = 2 -> ['ne26', 'ne27']
    Level crossing -> Lc06
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [1491, -120] | Coordinate: 0.5373
    Switches -> Sw12
        ContinueCourse -> right -> ne26

```

```

BranchCourse -> left -> ne27
Node ne23:
    Track = track3
    Neighbours = 2 -> ['ne25', 'ne27']
    Level crossing -> Lc07
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [-449, 300] | Coordinate: 0.3273
    Level crossing -> Lc09
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [311, 300] | Coordinate: 0.8148
    Switches -> Sw11
        ContinueCourse -> right -> ne25
        BranchCourse -> left -> ne27
Node ne25:
    Track = track4
    Neighbours = 2 -> ['ne23', 'ne27']
    Level crossing -> Lc05
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [1211, 300] | Coordinate: 0.4248
Node ne26:
    Track = track1
    Neighbours = 2 -> ['ne3', 'ne27']
    Level crossing -> Lc03
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [-449, -120] | Coordinate: 0.7421
    Level crossing -> Lc04
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [527, -120] | Coordinate: 0.2487
Node ne27:
    Track = track5
    Neighbours = 4 -> ['ne3', 'ne23', 'ne25', 'ne26']

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código G.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código G.4: SafePoints.RNA

```

ne3:
    Next: [[1336, -120]]
    Prev: [[1736, -120]]
ne23:

```

```

    Next: [[-360, 300], [-694, 300], [66, 300]]
    Prev: [[40, 300], [-294, 300], [466, 300]]
ne25:
    Next: [[966, 300]]
    Prev: [[1366, 300]]
ne26:
    Next: [[-241, -120], [-604, -120], [372, -120]]
    Prev: [[159, -120], [-204, -120], [772, -120]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura G.3.

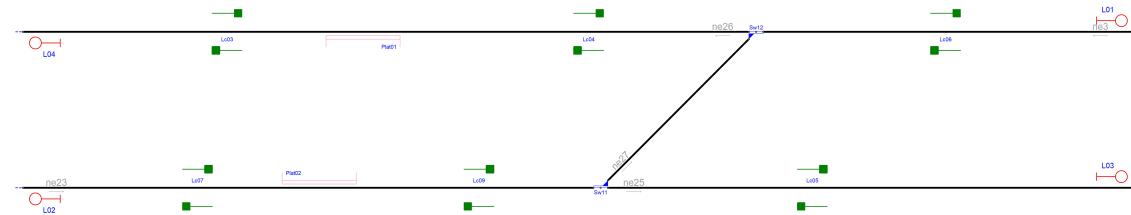


Figura G.3: Señalamiento generado por el RNA para proteger el fin de vía.

Al no existir finales de vías absolutos, el RNA no les asignó señalamiento. En cambio, los finales de vías relativos poseen las señales de parada L01, L02, L03 y L04, cercanos al límite del externo del *netElement* al que pertenecen. La Figura G.4 no presenta cambios en el señalamiento, al no existir junturas entre los rieles que proteger.

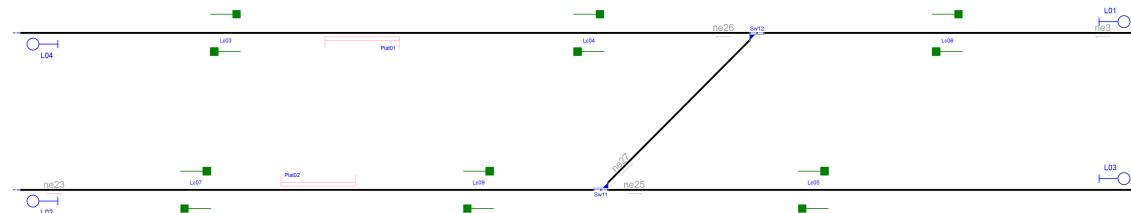


Figura G.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos, tal como sucede con los elementos LevelCrossing03 y Platform01. El señalamiento generado para proteger las plataformas y los cruces de vía, producto de aplicar el Algoritmo 9 y el Algoritmo 10, respectivamente, se ilustra en rojo en la Figura G.5. Las señales generadas

para proteger las plataformas son las señales de partida P17, P18, P19 y P20, mientras que las señales que protegen los cruces de vía son todas las señales entre X05 y X16.

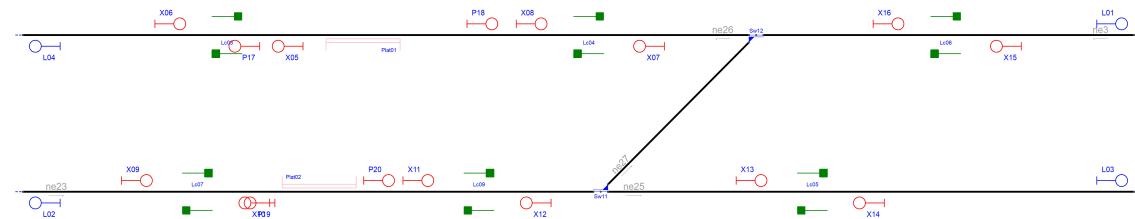


Figura G.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA genera las señales S22, C21 y H23 para proteger el cambio de vías Sw11 y las señales S25, C24 y H26 para proteger el cambio de vías Sw12. Las señales mencionadas se encuentran resaltadas en rojo en la Figura G.6.

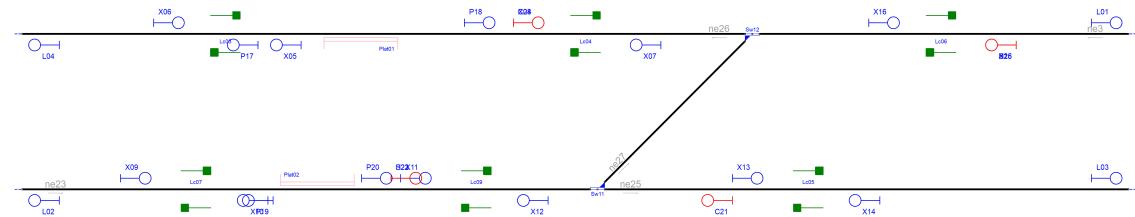


Figura G.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías Sw11 y Sw12, desplazando las señales hasta convertirlas en las señales H33 y H36 respectivamente.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: X11, H23, P18, X08, H26, X15, S19, P17 y X10. Las señales P20, X11 y H23 fueron eliminadas por su cercanía con la señal S22, con la cual comparten dirección y sentido. Lo mismo ocurre entre las señales H26 y X15. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código G.5.

Código G.5: Reducción de señalamiento por prioridad de señales

```

Reducing redundant signals
removing sig17 for sig05
removing sig18 for sig08
removing sig08 for sig24
removing sig10 for sig19
removing sig11 for sig22
removing sig21 for sig14
  
```

```

removing sig15 for sig25
removing sig20 for sig22
removing sig23 for sig22
removing sig26 for sig25

```

El resultado de la simplificación del señalamiento se ilustra en la Figura G.7.

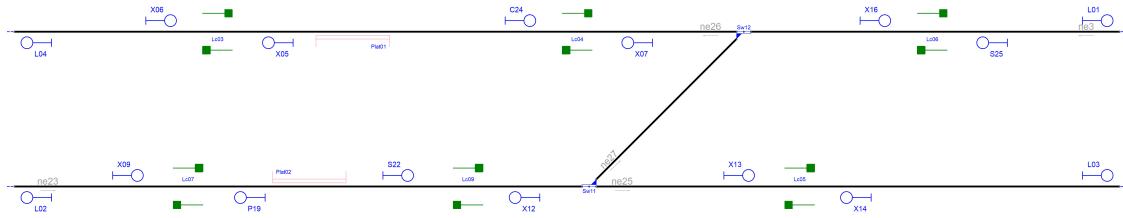


Figura G.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código 4.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código G.6: Signalling.RNA

```

sig01 [L01] >>:
    From: ne3 | To: lb70_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1940, -120] | Coordinate: 0.9019
sig02 [L02] <<:
    From: ne23 | To: oe178_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-860, 300] | Coordinate: 0.0641
sig03 [L03] >>:
    From: ne25 | To: oe179_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1940, 300] | Coordinate: 0.9305
sig04 [L04] <<:
    From: ne26 | To: lb69_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [-860, -120] | Coordinate: 0.0505
sig05 [X05] <<:
    From: ne26 | To: ne26_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [-204, 120] | Coordinate: 0.4005
sig06 [X06] >>:
    From: ne26 | To: ne26_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-604, 120] | Coordinate: 0.2168
sig07 [X07] <<:
    From: ne26 | To: ne26_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [772, 120] | Coordinate: 0.8831

```

```

sig09 [X09] >>:
    From: ne23 | To: ne23_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [-694, -300] | Coordinate: 0.4207
sig12 [X12] <<:
    From: ne23 | To: ne23_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [466, -300] | Coordinate: 0.9917
sig13 [X13] >>:
    From: ne25 | To: ne25_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [966, -300] | Coordinate: 0.4880
sig14 [X14] <<:
    From: ne25 | To: ne25_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1366, -300] | Coordinate: 0.6757
sig16 [X16] >>:
    From: ne3 | To: ne3_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [1336, 120] | Coordinate: 0.3890
sig19 [P19] <<:
    From: ne23 | To: ne23_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [-280, -300] | Coordinate: 0.5813
sig22 [S22] >>:
    From: ne23 | To: ne23_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [40, -300] | Coordinate: 0.7475
sig24 [C24] >>:
    From: ne26 | To: ne26_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [372, 120] | Coordinate: 0.6835
sig25 [S25] <<:
    From: ne3 | To: ne3_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [1736, 120] | Coordinate: 0.7403

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código G.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código G.7: Routes.RNA

```

route_1 [X05 << L04]:
    Path: ['ne26']
    LevelCrossings: ['Lc03']
route_2 [X06 >> C24]:
    Path: ['ne26']
    Platforms: ['Plat01']

```

```

    LevelCrossings: ['Lc03']
route_3 [X07 << X05]:
    Path: ['ne26']
    Platforms: ['Plat01']
    LevelCrossings: ['Lc04']
route_4 [X09 >> S22]:
    Path: ['ne23']
    Platforms: ['Plat02']
    LevelCrossings: ['Lc07']
route_5 [X12 << P19]:
    Path: ['ne23']
    Platforms: ['Plat02']
    LevelCrossings: ['Lc09']
route_6 [X13 >> L03]:
    Path: ['ne25']
    LevelCrossings: ['Lc05']
route_7 [X14 << X12]:
    Path: ['ne25', 'ne23']
    Switches: ['Sw11_N']
    LevelCrossings: ['Lc05']
route_8 [X16 >> L01]:
    Path: ['ne3']
    LevelCrossings: ['Lc06']
route_9 [P19 << L02]:
    Path: ['ne23']
    LevelCrossings: ['Lc07']
route_10 [S22 >> X13]:
    Path: ['ne23', 'ne25']
    Switches: ['Sw11_N']
    LevelCrossings: ['Lc09']
route_11 [S22 >> X16]:
    Path: ['ne23', 'ne27', 'ne3']
    Switches: ['Sw11_R', 'Sw12_R']
    LevelCrossings: ['Lc09']
route_12 [C24 >> X16]:
    Path: ['ne26', 'ne3']
    Switches: ['Sw12_N']
    LevelCrossings: ['Lc04']
route_13 [S25 << X07]:
    Path: ['ne3', 'ne26']
    Switches: ['Sw12_N']
    LevelCrossings: ['Lc06']
route_14 [S25 << X12]:
    Path: ['ne3', 'ne27', 'ne23']
    Switches: ['Sw11_R', 'Sw12_R']
    LevelCrossings: ['Lc06']

```

G.4. Red de grafos generada por el RNA

La información exportada en el Código G.3 (Infrastructure.RNA) Código G.4 (SafePoint.RNA), Código G.6 (Signalling.RNA) y Código G.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura G.8.

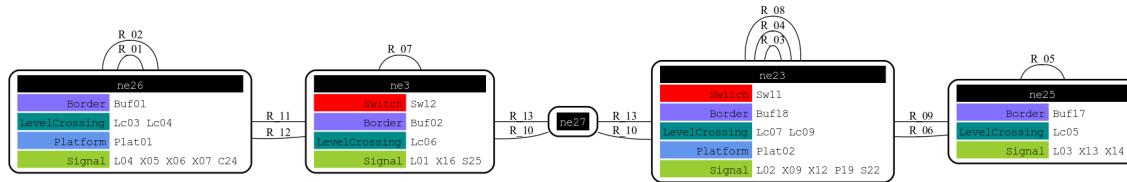


Figura G.8: Red de grafos generada por el RNA para el ejemplo 8.

Cada nodo del grafo de la Figura G.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

G.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código G.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla G.2.

Tabla G.2: Tabla de enclavamiento del ejemplo 8 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	X ₀₅	L ₀₄	-	-	Lc ₀₃	ne ₂₆
R ₀₂	X ₀₆	C ₂₄	-	Plat ₀₁	Lc ₀₃	ne ₂₆
R ₀₃	X ₀₇	X ₀₅	-	Plat ₀₁	Lc ₀₄	ne ₂₆
R ₀₄	X ₀₉	S ₂₂	-	Plat ₀₂	Lc ₀₇	ne ₂₃
R ₀₅	X ₁₂	P ₁₉	-	Plat ₀₂	Lc ₀₉	ne ₂₃
R ₀₆	X ₁₃	L ₀₃	-	-	Lc ₀₅	ne ₂₅
R ₀₇	X ₁₄	X ₁₂	Sw ₁₁ ^N	-	Lc ₀₅	ne ₂₅ -ne ₂₃
R ₀₈	X ₁₆	L ₀₁	-	-	Lc ₀₆	ne ₀₃
R ₀₉	P ₁₉	L ₀₂	-	-	Lc ₀₇	ne ₂₃
R ₁₀	S ₂₂	X ₁₃	Sw ₁₁ ^N	-	Lc ₀₉	ne ₂₃ -ne ₂₅
R ₁₁	S ₂₂	X ₁₆	Sw ₁₁ ^R +Sw ₁₂ ^R	-	Lc ₀₉	ne ₂₃ -ne ₀₃
R ₁₂	C ₂₄	X ₁₆	Sw ₁₂ ^N	-	Lc ₀₄	ne ₂₆ -ne ₀₃
R ₁₃	S ₂₅	X ₀₇	Sw ₁₂ ^N	-	Lc ₀₆	ne ₀₃ -ne ₂₆
R ₁₄	S ₂₅	X ₁₂	Sw ₀₄ ^R +Sw ₀₇ ^N	-	Lc ₀₆	ne ₀₃ -ne ₂₃

En una primera inspección podemos ver que el nuevo señalamiento tiene 14 rutas, versus las

12 rutas del señalamiento original (ver Tabla G.1). Esto se debe a que el RNA añadió paradas intermedias entre algunos elementos, dividiendo algunas rutas demasiado largas.

G.6. Validación del sistema

La validación de las rutas de la tabla de enclavamientos es realizada por el RNA aplicando el Algoritmo 16, explicado en la Sección 2.5. Las 12 rutas del señalamiento original (Tabla G.1) tienen 12 rutas equivalentes en el señalamiento generado por el RNA (Tabla G.2), tal como se puede visualizar en la Tabla G.3, generada automáticamente por el RNA.

Tabla G.3: Equivalencias entre las rutas originales y las generadas por el RNA.

Original	Señales	RNA	Señales
R ₀₁	S _{06-S₁₆}	R ₀₂	X _{06-C₂₄}
R ₀₂	S _{07-S₀₅}	R ₀₃	X _{07-X₀₅}
R ₀₃	S _{08-S₁₀}	R ₀₄	X _{09-S₂₂}
R ₀₄	S _{10-S₁₂}	R ₁₀	S _{22-X₁₃}
R ₀₅	S _{10-S₁₄}	R ₁₁	S _{22-X₁₆}
R ₀₆	S _{11-S₀₉}	R ₀₉	P _{19-L₀₂}
R ₀₇	S _{12-S₀₃}	R ₀₆	X _{13-L₀₃}
R ₀₈	S _{13-S₁₁}	R ₀₇	X _{14-X₁₂}
R ₀₉	S _{14-S₀₁}	R ₀₈	X _{16-L₀₁}
R ₁₀	S _{15-S₀₇}	R ₁₃	S _{25-X₀₇}
R ₁₁	S _{15-S₁₁}	R ₁₄	S _{25-X₁₂}
R ₁₂	S _{16-S₁₄}	R ₁₂	C _{24-X₁₆}

Las rutas R1 y R5 (Tabla 4.2) generadas por el RNA que no tienen equivalencias en el señalamiento original (Tabla 4.1) se deben a que el RNA creó señales extras. La ruta R1 es definida por el RNA al crear la señal X05 entre la plata forma Plat01 y el cruce de vías Lc03, lo cual genera una parada intermedia que antes no existía, hasta culminar la ruta en la señal L04. De la misma manera, el RNA define la ruta R5 al crear la señal P19, lo cual genera una parada intermedia entre la plataforma Plat02 y el cruce de vías Lc07. En definitiva, el RNA dividió las rutas R2 y R6 originales en las nuevas rutas R3+R1 y R5+R9. Lo cual mejora la logística de la red ferroviaria.

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.
- Principio de claridad (Algoritmo 18): rutas 100 % independientes.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.

- Principio de granularidad (Algoritmo 20): 100 % de rutas divididas a su mínima expresión.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

G.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura G.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código G.8.

Código G.8: Cantidad de elementos a implementar por el ACG

```
n_netElements:5
n_switch:2
n_doubleSwitch:0
n_borders:4
n_buffers:0
n_levelCrossings:6
n_platforms:2
n_scissorCrossings:0
n_signals:16
N : 42
```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 60 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 8 para automatizar la importación de los archivos del ejemplo 8 y desvincular cualquier otro conjunto de archivos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 42 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 42 en el Código G.8. El ejemplo 8 utiliza mas de 13150 sub módulos conectados automáticamente mediante mas de 25250 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla G.4. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos utilizados es baja y el tiempo de síntesis e implementación es de 38 y 53 segundos, respectivamente.

Tabla G.4: Síntesis e implementación del ejemplo 8 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	2234	2164	4.20-4.07 %
FF	2281	2284	2.14-2.15 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Apéndice H

Ejemplo 9

H.1. Topología ferroviaria original

El noveno y último ejemplo, ilustrado en la Figura H.1, es una topología de una estación real de Argentina. Presenta una vía principal con doble desvío, utilizado para maniobras y para el ascenso y descenso de los pasajeros en alguna de las tres plataformas centrales. Las vías son atravesadas perpendicularmente por dos calles, representadas por los seis cruces de vías. Todos los cambios de vías son simples, pero anidados, para permitir acceder a las vías superiores.

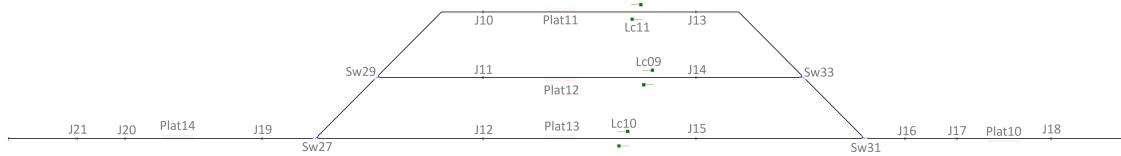


Figura H.1: Topología ferroviaria del ejemplo 9 sin señalamiento.

La topología presenta varias junturas entre los rieles, ya que el sistema original presentaba circuitos de vías y contadores de ejes, que fueron removidos para este ejercicio. No obstante, se mantienen las junturas para incrementar la dificultad del análisis.

H.2. Señalamiento original

El señalamiento original, ilustrado en la Figura H.2, no incluye señales al ser un ejemplo de diseño de una red ferroviaria para una locación real, de la cual no fue proporcionada ninguna información salvo la topología de la red. El objetivo de este ejemplo fue comparar el señalamiento generado sin tener ninguna otra referencia externa.



Figura H.2: Señalamiento original del ejemplo 9.

La cantidad de rutas existentes es indefinida, así como también el sentido de circulación de las vías.

H.3. Generación de señalamiento paso a paso

Inicialmente, el RNA ejecuta el Algoritmo 1 (ver Sección 2.1.3) para detectar todos los *netElements*, sus coordenadas iniciales y finales en la topología, y el sentido en el que fueron definidas. Al concluir el Algoritmo 1, el RNA ejecuta el Algoritmo 2 (ver Sección 2.1.3) para analizar la conexidad de la red. El resultado obtenido se muestra en el Código H.1, donde se describen las coordenadas de cada *netElement* y se confirma que la red es conexa.

Código H.1: Detección de *netElements* por parte del RNA

```
##### Starting Railway Network Analyzer #####
Reading .railML file
Creating railML object
Analysing railML object
Analysing graph
ne3 [9384, 0] [7584, 0] <<
ne40 [1680, 0] [3796, 0] >>
ne46 [3796, 0] [7584, 0] >>
ne48 [4216, 420] [3796, 0] <<
ne49 [4216, 420] [7164, 420] >>
ne50 [7164, 420] [4216, 420] <<
ne53 [7164, 420] [7584, 0] >>
The network is connected
```

Por ejemplo, el *netElement* ne50 inicia en la coordenada (7164;420) y finaliza en la coordenada (4216;420). El símbolo << indica que ne50 se encuentra definido de derecha a izquierda, ya que la componente x de la coordenada final es menor a la de la coordenada inicial, teniendo la misma componente y. Además, se puede comprobar que la lista obtenida es consistente con la Figura H.2. Por ejemplo, ne48, ne49 y ne50 comparten la coordenada (4216;420), que coincide con la coordenada del cambio de vías Sw29.

A continuación, el RNA detectará la infraestructura ferroviaria, las curvas peligrosas y los puntos medios de los netElements que el RNA considera demasiado largos. El análisis de la infraestructura se detalla en la Sección 2.1.5, Sección 2.1.6, Sección 2.1.7 y Sección 2.1.8, mientras que la detección de curvas y puntos medios se detalla en la Sección 2.1.4. El RNA ejecuta el Algoritmo 3, Algoritmo 4 y Algoritmo 5 para confirmar la detección de cambios de vías simples, dobles y en tijeras. El resultado de este proceso se puede visualizar en el Código H.2.

Código H.2: Detección de puntos críticos por parte del RNA

```
Analysing infrastructure --> Infrastructure.RNA
Detecting Danger --> Safe_points.RNA
ne3 has a RailJoint[J16] @ [7864, 0]
ne3 has a RailJoint[J17] @ [8219, 0]
ne3 has a RailJoint[J18] @ [8867, 0]
ne3 has a Platform[Plat10] @ [8542, 0]
ne40 has a RailJoint[J19] @ [3428, 0]
ne40 has a RailJoint[J20] @ [2485, 0]
ne40 has a RailJoint[J21] @ [2149, 0]
```

```

ne40 has a Platform[Plat14] @ [2848, 0]
ne46 has a RailJoint[J12] @ [4952, 0]
ne46 has a RailJoint[J15] @ [6424, 0]
ne46 has a Platform[Plat13] @ [5487, 0]
ne46 has a LevelCrossing[Lc10] @ [5915, 0]
ne49 has a RailJoint[J10] @ [4952, 870]
ne49 has a RailJoint[J13] @ [6424, 870]
ne49 has a Platform[Plat11] @ [5487, -870]
ne49 has a LevelCrossing[Lc11] @ [6006, -870]
ne49 has a Curve(3 lines) @ [[4666, 870], [6714, 870]]
ne50 has a RailJoint[J11] @ [4952, 420]
ne50 has a RailJoint[J14] @ [6424, 420]
ne50 has a Platform[Plat12] @ [5487, -420]
ne50 has a LevelCrossing[Lc09] @ [6096, -420]

```

Esta información es exportada por el RNA, con mayor detalle, en el archivo Infrastructure.RNA (Código H.3) que resume cada elemento ferroviario asociado a su respectivo *netElement*.

Código H.3: Infrastructure.RNA

```

Nodes: 7|Switches: 4|Signals: 0|Detectors: 12|Ends: 2|Barriers: 3
Node ne3:
    Track = track2
    TrainDetectionElements -> tde83
        Type -> insulatedRailJoint
    TrainDetectionElements -> tde84
        Type -> insulatedRailJoint
    TrainDetectionElements -> tde85
        Type -> insulatedRailJoint
    Type = BufferStop -> ['bus5']
    Neighbours = 2 -> ['ne46', 'ne53']
    Switches -> Sw31
        ContinueCourse -> left -> ne46
        BranchCourse -> right -> ne53
Node ne40:
    Track = track1
    TrainDetectionElements -> tde86
        Type -> insulatedRailJoint
    TrainDetectionElements -> tde87
        Type -> insulatedRailJoint
    TrainDetectionElements -> tde88
        Type -> insulatedRailJoint
    Type = BufferStop -> ['bus1']
    Neighbours = 2 -> ['ne46', 'ne48']
    Switches -> Sw27
        ContinueCourse -> right -> ne46
        BranchCourse -> left -> ne48
Node ne46:
    Track = track3
    TrainDetectionElements -> tde71
        Type -> insulatedRailJoint

```

```

TrainDetectionElements -> tde82
    Type -> insulatedRailJoint
    Neighbours = 4 -> ['ne3', 'ne40', 'ne48', 'ne53']
    Level crossing -> Lc10
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [5960, 0] | Coordinate: 0.5712
Node ne48:
    Track = track4
    Neighbours = 4 -> ['ne40', 'ne46', 'ne49', 'ne50']
    Switches -> Sw29
        ContinueCourse -> left -> ne49
        BranchCourse -> right -> ne50
Node ne49:
    Track = track5
    TrainDetectionElements -> tde69
        Type -> insulatedRailJoint
    TrainDetectionElements -> tde80
        Type -> insulatedRailJoint
    Neighbours = 3 -> ['ne48', 'ne50', 'ne53']
    Level crossing -> Lc11
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [6051, -870] | Coordinate: 0.6086
Node ne50:
    Track = track6
    TrainDetectionElements -> tde70
        Type -> insulatedRailJoint
    TrainDetectionElements -> tde81
        Type -> insulatedRailJoint
    Neighbours = 3 -> ['ne48', 'ne49', 'ne53']
    Level crossing -> Lc09
        Protection -> true | Barriers -> none | Lights -> none Acoustic ->
            ↛ none
        Position -> [6051, -420] | Coordinate: 0.3776
Node ne53:
    Track = track7
    Neighbours = 4 -> ['ne3', 'ne46', 'ne49', 'ne50']
    Switches -> Sw33
        ContinueCourse -> right -> ne49
        BranchCourse -> left -> ne50

```

La información de la infraestructura es utilizada por el RNA para detectar los puntos críticos de la red, es decir, las zonas donde es recomendable colocar una señal, según el sentido de circulación que se desee. El resultado es exportado al archivo SafePoints.RNA (Código H.4). En el caso de que un mismo *netElement* tenga más de un punto crítico para el mismo sentido, el RNA tomará el más cercano al elemento a proteger. El criterio de selección de puntos críticos se aplica para cada elemento ferroviario detectado, cada curva y cada cambio de vías y fue explicado en las secciones correspondientes ya mencionadas.

Código H.4: SafePoints.RNA

```

ne3:
    Next: [[7764.0, 0], [8119.0, 0], [8767.0, 0]]
    Prev: [[7964.0, 0], [8319.0, 0], [8967.0, 0]]
ne40:
    Next: [[3328.0, 0], [2385.0, 0], [2049.0, 0]]
    Prev: [[3528.0, 0], [2585.0, 0], [2249.0, 0]]
ne46:
    Next: [[4852.0, 0], [6324.0, 0], [5715, 0]]
    Prev: [[5052.0, 0], [6524.0, 0], [6115, 0]]
ne49:
    Next: [[4852.0, 870], [6324.0, 870], [5806, -870], [6614.0, 870]]
    Prev: [[5052.0, 870], [6524.0, 870], [6206, -870], [4766.0, 870]]
ne50:
    Next: [[4852.0, 420], [6324.0, 420], [5896, -420]]
    Prev: [[5052.0, 420], [6524.0, 420], [6296, -420]]

```

Una vez que el RNA detectó cada punto crítico de la red ferroviaria, procede a generar el señalamiento. El orden en que se procesan los elementos ferroviarios no impacta en el resultado final, pero para poder describirlo de forma ordenada se iniciará generando el señalamiento para proteger los finales de vías, las junturas entre rieles, las plataformas (explicado en la Sección 2.2.3), los cruces de vía (explicado en la Sección 2.2.4) y los cambios de vías (explicado en la Sección H.3). Luego se procederá a mostrar el señalamiento completo antes y después de la simplificación de señales (explicado en la Sección).

Tal como se explicó en la Sección 2.2.1, el RNA aplica el Algoritmo 6 y el Algoritmo 7 para generar las señales para proteger los finales de vías relativos y absolutos. Estas señales son ilustradas en la Figura H.3.



Figura H.3: Señalamiento generado por el RNA para proteger el fin de vía.

Los finales de vías absolutos son protegidos por las señales de parada T01 y T03, y las señales de partida son T02 y T04. A su vez, al no existir los finales de vías relativos, el RNA no les asignó ninguna señal para su protección.

La Figura H.4 ilustra la generación de señales destinadas a proteger las junturas entre los rieles. Estas señales se obtuvieron al aplicar el Algoritmo 8, tal como fue explicado en la Sección 2.2.2. Las señales generadas son todas las señales entre J05 y J28, indicadas en color rojo.



Figura H.4: Señalamiento generado por el RNA para proteger las junturas.

Al generar el señalamiento para proteger la infraestructura, tal como se explicó en la Sección 2.3.2, el Algoritmo 15 simplificará las señales entre dos elementos ferroviarios si no existe espacio suficiente entre ellos, tal como sucede con los elementos LevelCrossing09 y Platform12. El señalamiento generado para proteger las plataformas y los cruces de vía se ilustra en rojo en la Figura H.5. Las señales generadas para proteger las plataformas no fueron generadas, al encontrarse próximas a las señales de protección de junturas, por lo que el RNA decidió no superponer ambas señales. Mientras que las señales que protegen los cruces de vía son las señales X29 a X34.

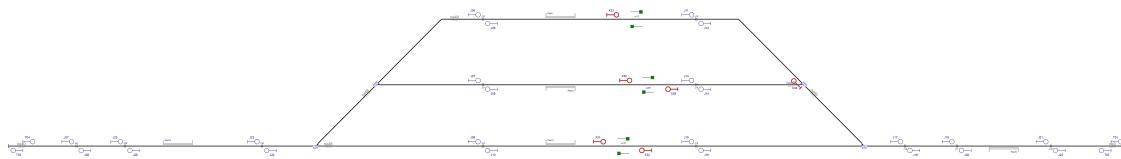


Figura H.5: Señalamiento generado por el RNA para proteger plataformas y cruces de vía.

Al aplicar el Algoritmo 11 de generación de señalamiento para cambios de vías, tal como fue explicado en la Sección , el RNA genera las señales S36 , C35, H36 y H38 para proteger el cambio de vías Sw27; las señales S42, C41, H43 y H44 para proteger el cambio de vías Sw31; las señales C39 y B46 para proteger el cambio de vías Sw29 y la señal C45 para proteger el cambio de vías Sw33. Las señales mencionadas se encuentran resaltadas en rojo en la Figura H.6.

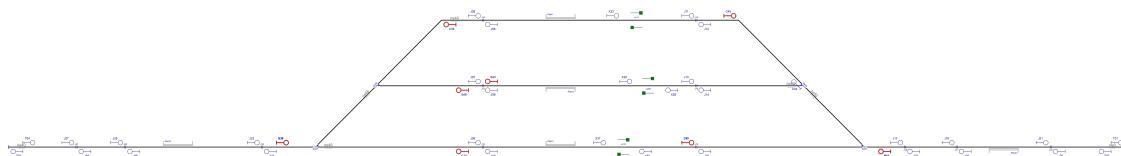


Figura H.6: Señalamiento generado por el RNA para proteger los cambios de vías.

Una vez obtenido todo el señalamiento, el RNA procede a simplificar las señales redundantes, repetidas o cuyas funciones o ubicaciones se superponen entre sí. El proceso de simplificación de señales fue explicado en la Sección H.3. El Algoritmo 15 de herencia vertical fue aplicado en las señales B entre los cambios de vías Sw31 y Sw33, desplazando las señales hasta convertirlas en las señales H43 y H respectivamente. Análogamente, entre los cambios de vías Sw27 y Sw29 se convirtieron en las señales H37 y H38 respectivamente.

Las señales simplificadas al aplicar el Algoritmo 15 de herencia horizontal son: J22, J28, J27, C39, J08, C35, C45, X34, X30, X29, J15, X32, J19, J20, J18, J23, H37, H38, B40, H43 y H44. Las

señales J23, H37 y H38 fueron eliminadas por su cercanía con la señal S36, con la cual comparten dirección y sentido. Lo mismo ocurre entre el par de señales H43/H44 y la señal S42; además de varios otros casos de simplificaciones simples. En todos los casos, se aplicó el Algoritmo 15, diseñado para agrupar objetos cercanos como un único objeto, generando el señalamiento acorde a los elementos contenidos en cada extremo del nuevo elemento contenedor.

Finalmente, las señales son simplificadas aplicando el Algoritmo 14 de eliminación por prioridad de señales. El resultado de este proceso es detallado en el Código H.5.

Código H.5: Reducción de señalamiento por prioridad de señales

```
Reducing redundant signals
removing J22 for T02
removing J28 for T03
removing J27 for T04
removing C39 for J06
removing J08 for B40
removing C35 for J10
removing C45 for J11
removing X34 for J12
removing X30 for J13
removing X29 for J14
removing J15 for C41
removing X32 for J16
removing J19 for J17
removing J20 for J18
removing J18 for S42
removing J23 for S36
removing H37 for S36
removing H38 for S36
removing B40 for B46
removing H43 for S42
removing H44 for S42
```

El resultado de la simplificación del señalamiento se ilustra en la Figura H.7.

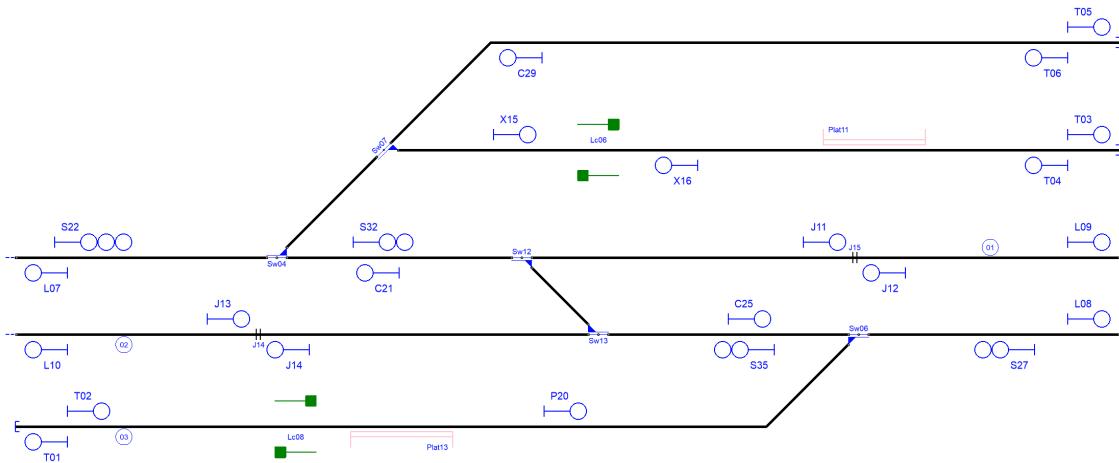


Figura H.7: Señalamiento generado y simplificado por el RNA.

Además, toda la información del señalamiento generado es exportada por el RNA en el archivo Signalling.RNA (Código 4.6), que incluye información detallada de la posición, orientación, sentido, coordenada, nombre y tipo de señal.

Código H.6: Signalling.RNA

```

T01 [T01] >:
  From: ne3 | To: bus5_right
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [9284, 0] | Coordinate: 0.9444
T02 [T02] <:
  From: ne3 | To: ne3_left
  Type: Stop | Direction: normal | AtTrack: left
  Position: [9284, 0] | Coordinate: 0.9444
T03 [T03] <:
  From: ne40 | To: bus1_left
  Type: Stop | Direction: reverse | AtTrack: right
  Position: [1780, 0] | Coordinate: 0.0472
T04 [T04] >:
  From: ne40 | To: ne40_right
  Type: Stop | Direction: normal | AtTrack: left
  Position: [1780, 0] | Coordinate: 0.0472
J05 [J05] >:
  From: ne49 | To: ne49_right
  Type: Circulation | Direction: normal | AtTrack: left
  Position: [4852.0, -870] | Coordinate: 0.4392
J06 [J06] <:
  From: ne49 | To: ne49_left
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [5052.0, -870] | Coordinate: 0.4995
J07 [J07] >:
  From: ne50 | To: ne50_right
  Type: Circulation | Direction: reverse | AtTrack: right
  Position: [4852.0, -420] | Coordinate: 0.2157

```

```

J09 [J09] >:
    From: ne46 | To: ne46_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [4852.0, 0] | Coordinate: 0.2787
J10 [J10] <:
    From: ne46 | To: ne46_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [5052.0, 0] | Coordinate: 0.3315
J11 [J11] >:
    From: ne49 | To: ne49_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [6324.0, -870] | Coordinate: 0.8825
J12 [J12] <:
    From: ne49 | To: ne49_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [6524.0, -870] | Coordinate: 0.9427
J13 [J13] >:
    From: ne50 | To: ne50_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [6324.0, -420] | Coordinate: 0.7150
J14 [J14] <:
    From: ne50 | To: ne50_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [6524.0, -420] | Coordinate: 0.7829
J16 [J16] <:
    From: ne46 | To: ne46_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [6524.0, 0] | Coordinate: 0.7201
J17 [J17] >:
    From: ne3 | To: ne3_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [7764.0, 0] | Coordinate: 0.1
J21 [J21] >:
    From: ne3 | To: ne3_right
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [8767.0, 0] | Coordinate: 0.6572
J24 [J24] <:
    From: ne40 | To: ne40_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [3528.0, 0] | Coordinate: 0.8733
J25 [J25] >:
    From: ne40 | To: ne40_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [2385.0, 0] | Coordinate: 0.3331
J26 [J26] <:
    From: ne40 | To: ne40_left
    Type: Circulation | Direction: reverse | AtTrack: right
    Position: [2585.0, 0] | Coordinate: 0.4276
X31 [X31] >:
    From: ne46 | To: ne46_right
    Type: Circulation | Direction: normal | AtTrack: left

```

```

Position: [5715, 0] | Coordinate: 0.5065
X33 [X33] >>:
    From: ne49 | To: ne49_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [5806, 870] | Coordinate: 1.0096
S36 [S36] >>:
    From: ne40 | To: ne40_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [3528.0, 0] | Coordinate: 0.8733
C41 [C41] >>:
    From: ne46 | To: ne46_right
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [6324.0, 0] | Coordinate: 0.6673
S42 [S42] <<:
    From: ne3 | To: ne3_left
    Type: Circulation | Direction: normal | AtTrack: left
    Position: [7764.0, 0] | Coordinate: 0.1
B46 [B46] <<:
    From: ne50 | To: ne50_right
    Type: Manouver | Direction: normal | AtTrack: left
    Position: [4852.0, -420] | Coordinate: 0.2157

```

Al finalizar la generación del señalamiento, el RNA ejecuta el Algoritmo 15, explicado en la Sección 2.4, para detectar todas las posibles rutas admitidas por la red para crear la tabla de enclavamientos. La cuál puede ser visualizada en el archivo Routes.RNA (Código H.7). La misma detalla las señales de inicio y final, los *netElements* abarcados por la ruta y cualquier infraestructura involucrada, incluyendo el estado que deben tener para que la ruta sea activada.

Código H.7: Routes.RNA

```

route_1 [T02 << S42]:
    Path: ['ne3']
    Platforms: ['Plat10']
route_2 [T04 >> J25]:
    Path: ['ne40']
route_3 [J05 >> J11]:
    Path: ['ne49']
    Platforms: ['Plat11']
    LevelCrossings: ['Lc11']
route_4 [J06 << J24]:
    Path: ['ne49', 'ne48', 'ne40']
    Switches: ['Sw27_R', 'Sw29_N']
route_5 [J07 >> J13]:
    Path: ['ne50']
    Platforms: ['Plat12']
    LevelCrossings: ['Lc09']
route_6 [J09 >> X31]:
    Path: ['ne46']
    Platforms: ['Plat13']
route_7 [J10 << J24]:
    Path: ['ne46', 'ne40']

```

```

        Switches: ['Sw27_N']
route_8 [J11 >> J17]:
    Path: ['ne49', 'ne53', 'ne3']
    Switches: ['Sw31_R', 'Sw33_N']
route_9 [J12 << J06]:
    Path: ['ne49']
    Platforms: ['Plat11']
    LevelCrossings: ['Lc11']
route_10 [J13 >> J17]:
    Path: ['ne50', 'ne53', 'ne3']
    Switches: ['Sw31_R', 'Sw33_R']
route_11 [J14 << B46]:
    Path: ['ne50']
    Platforms: ['Plat12']
    LevelCrossings: ['Lc09']
route_12 [J16 << J10]:
    Path: ['ne46']
    Platforms: ['Plat13']
    LevelCrossings: ['Lc10']
route_13 [J17 >> J21]:
    Path: ['ne3']
    Platforms: ['Plat10']
route_14 [J21 >> T01]:
    Path: ['ne3']
route_15 [J24 << J26]:
    Path: ['ne40']
    Platforms: ['Plat14']
route_16 [J25 >> S36]:
    Path: ['ne40']
    Platforms: ['Plat14']
route_17 [J26 << T03]:
    Path: ['ne40']
route_18 [X31 >> C41]:
    Path: ['ne46']
    LevelCrossings: ['Lc10']
route_19 [X33 >> J17]:
    Path: ['ne49', 'ne53', 'ne3']
    Switches: ['Sw31_R', 'Sw33_N']
    LevelCrossings: ['Lc11']
route_20 [S36 >> J09]:
    Path: ['ne40', 'ne46']
    Switches: ['Sw27_N']
route_21 [S36 >> J05]:
    Path: ['ne40', 'ne48', 'ne49']
    Switches: ['Sw27_R', 'Sw29_N']
route_22 [S36 >> J07]:
    Path: ['ne40', 'ne48', 'ne50']
    Switches: ['Sw27_R', 'Sw29_R']
route_23 [C41 >> J17]:
    Path: ['ne46', 'ne3']
    Switches: ['Sw31_N']

```

```

route_24 [S42 << J16]:
    Path: ['ne3', 'ne46']
    Switches: ['Sw31_N']
route_25 [S42 << J12]:
    Path: ['ne3', 'ne53', 'ne49']
    Switches: ['Sw31_R', 'Sw33_N']
route_26 [S42 << J14]:
    Path: ['ne3', 'ne53', 'ne50']
    Switches: ['Sw31_R', 'Sw33_R']
route_27 [B46 << J24]:
    Path: ['ne50', 'ne48', 'ne40']
    Switches: ['Sw27_R', 'Sw29_R']

```

H.4. Red de grafos generada por el RNA

La información exportada en el Código H.3 (Infrastructure.RNA) Código H.4 (SafePoint.RNA), Código H.6 (Signalling.RNA) y Código H.7 (Routes.RNA) es resumida de forma gráfica por el RNA para una mejor interpretación. El resultado de este resumen se ilustra en el diagrama de la Figura H.8.

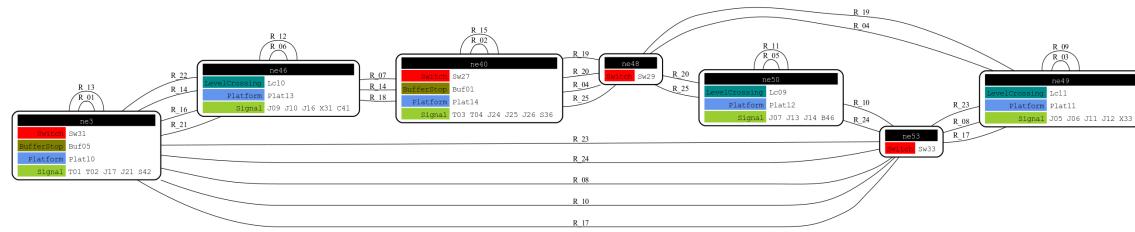


Figura H.8: Red de grafos generada por el RNA para el ejemplo 9.

Cada nodo del grafo de la Figura H.8 corresponde a un *netElement*. En cada nodo se listan todos los elementos ferroviarios contenidos por en *netElement*. Las aristas del grafo son las rutas que los conectan. De esta manera, es posible detectar visualmente cualquier nodo aislado de la red o nodos que solo son accedidos en un sentido. Por ejemplo, si entre dos nodos no existe una cantidad par de rutas, entonces solamente se puede circular entre esos nodos en un solo sentido.

H.5. Señalamiento generado por el RNA

El RNA también exporta la misma información mostrada en el Código H.8 en una hoja de cálculo, similar a la que se visualiza en la Tabla H.1.

Tabla H.1: Tabla de enclavamiento del ejemplo 9 generada por el RNA.

Ruta	Inicio	Final	Cambio	Plataforma	Cruce	netElement
R ₀₁	T ₀₂	S ₄₂	-	Plat ₁₀	-	ne ₃ -ne ₃
R ₀₂	T ₀₄	J ₂₅	-	-	-	ne ₄₀ -ne ₄₀
R ₀₃	J ₀₅	J ₁₁	-	Plat ₁₁	Lc ₁₁	ne ₄₉ -ne ₄₉
R ₀₄	J ₀₆	J ₂₄	Sw ₂₇ ^R +Sw ₂₉ ^N	-	-	ne ₄₉ -ne ₄₀
R ₀₅	J ₀₇	J ₁₃	-	Plat ₁₂	Lc ₀₉	ne ₅₀ -ne ₅₀
R ₀₆	J ₀₉	X ₃₁	-	Plat ₁₃	-	ne ₄₆ -ne ₄₆
R ₀₇	J ₁₀	J ₂₄	Sw ₂₇ ^N	-	-	ne ₄₆ -ne ₄₀
R ₀₈	J ₁₁	J ₁₇	Sw ₃₁ ^R +Sw ₃₃ ^N	-	-	ne ₁₄₉ -ne ₀₃
R ₀₉	J ₁₂	J ₀₆	-	Plat ₁₁	Lc ₁₁	ne ₄₉ -ne ₄₉
R ₁₀	J ₁₃	J ₁₇	Sw ₃₁ ^N +Sw ₃₃ ^R	-	-	ne ₅₀ -ne ₀₃
R ₁₁	J ₁₄	B ₄₆	-	Plat ₁₂	Lc ₀₉	ne ₅₀ -ne ₅₀
R ₁₂	J ₁₆	J ₁₀	-	Plat ₁₃	Lc ₁₀	ne ₄₆ -ne ₄₆
R ₁₃	J ₁₇	J ₂₁	-	Plat ₁₀	-	ne ₀₃ -ne ₀₃
R ₁₄	J ₂₁	T ₀₁	-	-	-	ne ₀₃ -ne ₀₃
R ₁₅	J ₂₄	J ₂₆	-	Plat ₁₄	-	ne ₄₀ -ne ₄₀
R ₁₆	J ₂₅	S ₃₆	-	Plat ₁₄	-	ne ₄₀ -ne ₄₀
R ₁₇	J ₂₆	T ₀₃	-	-	-	ne ₄₀ -ne ₄₀
R ₁₈	X ₃₁	C ₄₁	-	-	Lc ₁₀	ne ₄₆ -ne ₄₆
R ₁₉	X ₃₃	J ₁₇	Sw ₃₁ ^R +Sw ₃₃ ^N	-	Lc ₁₁	ne ₄₉ -ne ₀₃
R ₂₀	S ₃₆	J ₀₉	Sw ₂₇ ^N	-	-	ne ₄₀ -ne ₄₆
R ₂₁	S ₃₆	J ₀₅	Sw ₂₇ ^R +Sw ₂₉ ^N	-	-	ne ₄₀ -ne ₄₉
R ₂₂	S ₃₆	J ₀₇	Sw ₂₇ ^R +Sw ₂₉ ^R	-	-	ne ₄₀ -ne ₅₀
R ₂₃	C ₄₁	J ₁₇	Sw ₃₁ ^N	-	-	ne ₄₆ -ne ₀₃
R ₂₄	S ₄₂	J ₁₆	Sw ₃₁ ^N	-	-	ne ₀₃ -ne ₄₆
R ₂₅	S ₄₂	J ₁₂	Sw ₃₁ ^R +Sw ₃₃ ^N	-	-	ne ₀₃ -ne ₄₉
R ₂₆	S ₄₂	J ₁₄	Sw ₃₁ ^R +Sw ₃₃ ^R	-	-	ne ₀₃ -ne ₅₀
R ₂₇	B ₄₆	J ₂₄	Sw ₂₇ ^R +Sw ₂₉ ^R	-	-	ne ₅₀ -ne ₄₀

En una primera inspección podemos ver que el nuevo señalamiento tiene 27 rutas. Las mismas cubren toda la topología en ambas direcciones, permitiendo todos los movimientos posibles. Además, todas las rutas inician y terminan en lugares críticos: antes de una bifurcación o convergencia de vías, antes de una plataforma, antes de un cruce de vías o antes de una juntura. Lo cual permite al conductor anticiparse a los próximos posibles peligros.

H.6. Validación del sistema de enclavamientos

Para finalizar, el RNA comprueba los principios de señalamiento ferroviario explicados en la Sección , aplicando los algoritmos indicados, de los cuáles se obtuvieron los siguientes resultados:

- Principio de autoridad (Algoritmo 17): cobertura del 100 % de los *netElements*.

- Principio de claridad (Algoritmo 18): Al no existir un señalamiento original, no es posible realizar una comparación ruta a ruta entre ambos sistemas.
- Principio de anticipación (Algoritmo 19): cobertura del 100 % de los puntos críticos.
- Principio de granularidad (Algoritmo 20): Al no existir un señalamiento original, no es posible realizar una comparación ruta a ruta entre ambos sistemas.
- Principio de terminalidad (Algoritmo 21): 100 % de finales de vías protegidos.
- Principio de infraestructura (Algoritmo 22): 100 % de infraestructura protegida.
- Principio de no bloqueo (Algoritmo 23): 100 % de cambios de vías protegidos.

H.7. Sistema generado por el ACG

En base a la red de grafos, ilustrada en la Figura H.8, el ACG determinó la siguiente cantidad de elementos, tal puede visualizarse en el Código H.8.

Código H.8: Cantidad de elementos a implementar por el ACG

```

n_netElements:7
n_switch:4
n_doubleSwitch:0
n_borders:0
n_buffers:2
n_levelCrossings:3
n_platforms:5
n_scissorCrossings:0
n_signals:25
N : 66

```

Se repetirán los pasos detallados en el ejemplo 1, Sección 4.1.7, por lo que solamente se destacarán aspectos particulares de este ejemplo. El ACG genera 84 archivos en formato VHDL. El archivo *Arty_Z7-10.XDC* es el mismo para todos los ejemplos, al ser invariante respecto a la cantidad de pines a utilizar. Se deberá modificar el script mostrado en el Código 4.10 y cambiar el parámetro *chosen* a 9 para automatizar la importación de los archivos del ejemplo 9 y desvincular cualquier otro conjunto de archivos de ejemplos anteriores.

Una vez ejecutado el script, Vivado ordenará los archivos de forma jerárquica, donde el módulo *global* incluye todos los módulos que fueron detallados en la Sección 3.2. Cada una de las instancias del módulo *network* contienen sus propias 66 instancias de los mismos módulos de cada elemento ferroviario ya que N, cantidad de elementos ferroviarios, es 66 en el Código H.8. El ejemplo 9 utiliza mas de 22900 sub módulos conectados automáticamente mediante mas de 42650 señales, lo cual se aleja bastante de un desarrollo que pueda realizarse manualmente de forma trivial.

Cuando Vivado genera el diagrama de bloques ya tenemos la certeza de que el código VHDL ha pasado la prueba de sintaxis del entorno de desarrollo. A continuación, se deberá sintetizar e implementar el sistema para generar el bitstream que será utilizado para programar la FPGA. Los procesos de síntesis e implementación fueron detallados en el ejemplo 1, Sección 4.1.7.

Los resultados de ambos procesos son detallados en la Tabla H.2. Los porcentajes de uso son calculados por Vivado automáticamente, teniendo en cuenta que la plataforma Arty Z7 20 posee 53200 Look-Up-Tables (LUTs), 106400 Flip-Flops (FFs), 125 Pines de entrada y salida (IOs) y 32 Buffers (BUFGs), tal como se explicó en la Sección 3.4. En este ejemplo, la cantidad de recursos

utilizados es baja y el tiempo de síntesis e implementación es de 50 y 1 minuto con 11 segundos, respectivamente.

Tabla H.2: Síntesis e implementación del ejemplo 9 generado por el ACG.

Recursos	Síntesis	Implementación	Uso
LUT	3857	3829	7.25-7.20 %
FF	4134	4137	3.89 %
IO	15	15	12.00 %
BUFG	3	3	9.38 %

Referencias

- [1] A. Iliasov, D. Taylor, L. Laibinis y A. Romanovsky, “Practical Verification of Railway Signalling Programs,” *IEEE Transactions on Dependable and Secure Computing*, 2022. DOI: 10.1109/TDSC.2022.3141555.
- [2] M. Heinrich et al., “Security Requirements Engineering in Safety-Critical Railway Signalling Networks,” *Security and Communication Networks*, 2018. DOI: 10.1155/2019/8348925.
- [3] R. Safety y S. B. (RSSB), “Signalling Principles Handbook: Interlocking Principles (Former Railway Group Standard GK/RT0060) - NR/L2/SIG/30009/GKRT0060,” 2020.
- [4] N. Goverment, “Rail Signalling Principles for Light Rail Networks,” *Transport Asest Standards Authority*, 2020.
- [5] R. I. Safety y S. B. (RISSB), “Signalling Principles,” 2018.
- [6] N. H. Konig y G. B. Raymond, “Standards for functions and interfaces as a basis for CBTC on mainline railways in Europe,” 2003.
- [7] M. A. Oz, O. T. Kaymakci y A. Koyun, “A safety related perspective for the power supply systems in railway industry,” *Eksplotacja i Niewazownosc - Maintenance and Reliability*, 2017. DOI: 10.17531/ein.2017.1.16.
- [8] H. Olsen y M. Owens, “SSI - Australian Railways,” *ARTC, Australian Rail Track Corporation LTD*, 2005.
- [9] R. Safety y S. B. (RSSB), “Block System Interface Requirements,” 2013.
- [10] “Reglamento Interno Técnico Operativo,” Trenes Argentinos. (1993), dirección: <https://www.argentina.gob.ar/sites/default/files/rito.pdf> (visitado 11-02-2024).
- [11] F. How. “Interlocking, back to basic.” (2022), dirección: <https://www.irse.org/Qualifications-Careers/Exam-Support-Resources/Module-A>.
- [12] “Fundamental Requirements for Train Control Systems,” Institution of Railway Signal Engineers. (2024), dirección: <https://www.irse.org/Publications-Resources/Misc-publications> (visitado 11-02-2024).
- [13] M. P. Lindegaard, P. Viuf y A. E. Haxthausen, “Modelling Railway Interlocking system,” *IFAC Control in Transportation Systems*, 2000. DOI: 10.1016/S1474-6670(17)38143-0.
- [14] “Normas para los cruces entre caminos y vias ferreas,” *Ministerio de Economía, Secretaría de Estado de Transporte y Obras Públicas de la República Argentina*,
- [15] A. Chiappini, A. Cimatti, L. Macchi y B. Vittorini, “Formalization and validation of a subset of the European Train Control System,” *ACM/IEEE 32nd International Conference on Software Engineering*, 2010. DOI: 10.1145/1810295.1810312.
- [16] L. Huang, “The Past, Present and Future of Railway Interlocking System,” *2020 IEEE 5th International Conference on Intelligent Transportation Engineering (ICITE)*, 2020. DOI: 10.1109/ICITE50838.2020.9231438.

- [17] S. Patalay, "Analysis of Railway Interlocking Systems,"
- [18] R. Cavada, A. Cimatti, S. Mover, M. Sessa, G. Cadavero y G. Scaglione, "Analysis of Relay Interlocking Systems via SMT-based Model Checking of Switched Multi-Domain Kirchhoff Networks," *Formal Methods in Computer Aided Design (FMCAD)*, 2018. doi: 10.23919/FMCAD.2018.8603007.
- [19] F. van Dijk, W. Fokkink, G. Kolk, P. van de Ven y B. van Vlijmen, "EURIS, a Specification Method for Distributed Interlockings," *Computer Safety, Reliability and Security*, 1998. doi: 10.1007/3-540-49646-7_23.
- [20] R. Dobias y H. Kubatova, "FPGA Based Desing of the Railway's Interlocking Equipments," *Euromicro Symposium on Digital System Design*, 2004. doi: 10.1109/DSD.2004.1333312.
- [21] A. Cribbens, "Solid-state interlocking (SSI): an integrated electronic signalling system for mainline railways," *IEE Proceedings B - Electric Power Applications*, 1987. doi: 10.1049/ip-b.1987.0024.
- [22] D. Lutovac y T. Lutovac, "Towards an universal computer interlocking system," 1998.
- [23] S. Raghapur, "Safety Integrity Level - an Overview for FPGA Engineers," 2018.
- [24] "Safety design on FPGA's using soft Lockstep Processors," 2016.
- [25] "Safe FPGA Design Practices Safe FPGA Design Practices for Instrumentation and for Instrumentation and Control in Nuclear Plants,"
- [26] B. F. Flesch, B. Brand, R. M. de Figueiredo, L. R. Prade y M. R. da Silva, "Proposal of a functional safety methodology applied to fault tolerance in FPGA applications," *Latin-American Test Symposium (LATW)*, 2016. doi: 10.1109/LATW.2016.7483331.
- [27] A. Hayek y J. Börcsök, "SRAM-Based FPGA Design Techniques for Safety Related Systems Conforming to IEC 61508," *International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, 2012. doi: 10.1109/ICTEA.2012.6462892.
- [28] "8 Reasons to Use FPGAs in IEC 61508 Functional Safety Applications," *Altera*, 2013.
- [29] "Developing Functional Safety Systems with TÜV-Qualified FPGAs," *Altera*, 2010.
- [30] "DO-254 for the FPGA Designer," *Xilinx*, 2012.
- [31] D. Sabena et al., "Reconfigurable High Performance Architectures: How much are they ready for safety-critical applications?" *IEEE European Test Symposium (ETS)*, 2014. doi: 10.1109/ETS.2014.6847820.
- [32] C. Bernardeschi, L. Cassano y A. Domenici, "SRAM-Based FPGA Systems for Safety-Critical Applications: A Survey on Design Standards and Proposed Methodologies," *Journal of Computer Science and Technology*, 2013. doi: 10.1007/s11390-015-1530-5.
- [33] "On Applying Software Development Best Practice to FPGAs in Safety-Critical Systems," 2000.
- [34] C. Bernardeschi, L. Cassano y A. Domenici, "SRAM-based FPGA Systems for Safety-Critical Applications: A Survey on Design Standards and Proposed Methodologies," *Journal of Computer Science and Technology*, 2015.
- [35] G. Corradi, R. Girardey y J. Becker, "Xilinx tools facilitate development of FPGA Applications for IEC61508," *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2012. doi: 10.1109/AHS.2012.6268669.
- [36] "Railway and Industry Business Expertise," *MIOS Elettronica*, 2020.

- [37] I. Bakhmach, V. Kharchenko, A. Siora, V. Sklyar y A. Andrashov, “Experience of I&C systems modernization using FPGA technology,” *American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface*, 2010.
- [38] I. Bakhmach, V. Kharchenko, A. Siora, V. Sklyar y V. Tokarev, “Advanced I&C systems for NPPS based on FPGA technology: European experience,” *International Conference on Nuclear Engineering*, 2009.
- [39] V. Sklyar, V. Kharchenko, A. Siora, S. Malokhatko, V. Golovir e Y. Beliy, “Reliability and Availability Analysis of FPGA-based Instrumentation and Control Systems,” *International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, 2011.
- [40] V. Kharchenko, V. Sklyar, O. Odarushchenko y A. Ivasuyk, “Fault-Injection Testing: FIT-Ability, Optimal Procedure and Tool for FPGA-Based Systems SIL Certification,” *East-West Design & Test Symposium (EWDTS)*, 2013. DOI: 10.1109/EWDTS.2013.6673129.
- [41] V. Kharchenko, A. Siora, V. Sklyar, A. Volkoviy y V. Bezsaliiy, “Multi-diversity versus common cause failures: FPGA-based multi-version NPP I&C systems,” *International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC & HMIT)*, 2010.
- [42] A. Siora, V. Sklyar, Y. Rozen, S. Vinogradskaya y S. Vinogradskaya, “Licensing principles of FPGA-based NPP I&C systems,” *International Conference on Nuclear Engineering*, 2009. DOI: 10.1115/ICONE17-75270.
- [43] A. Fernández-León, A. Pouponnot y S. Habinc, “ESA FPGA Task Force: Lessons Learned,” 2002. dirección: <https://api.semanticscholar.org/CorpusID:198956430>.
- [44] M. M. Nicolás, L. F. Santiago, G. R. Adrián, Á. Nicolás y L. Ariel, “FPGA implementation of a critical railway interlocking system,” *IEEE Latin America Transactions*, 2020. DOI: 10.1109/TLA.2019.9082240.
- [45] V. Kharchenko, “Diversity for Safety and Security of Embedded and Cyber Physical Systems: Fundamentals Review and Industrial Cases,” *Biennial Baltic Electronics Conference (BEC)*, 2016. DOI: 10.1109/BEC.2016.7743719.
- [46] “Isolation Design Flow for Xilinx 7 Series FPGAs or Zynq-7000 AP SoCs (ISE Tools),” 2015.
- [47] “Isolation Design Flow for Xilinx 7 Series FPGAs or Zynq-7000 AP SoCs (Vivado Tools),” 2016.
- [48] D. Macii, M. Avancini, L. Benciolini, S. Dalpez, M. Corra y R. Passerone, “Design of a Redundant FPGA-based Safety System for Railroad Vehicles,” *Euromicro Conference on Digital System Design*, 2014. DOI: 10.1109/DSD.2014.96.
- [49] E. Gracie, A. Hayek y J. Börcsök, “Evaluation of FPGA Design Tools for Safety Systems with On-Chip Redundancy Referring to the Standard IEC 61508,” *International Conference on System Reliability and Safety (ICRS)*, 2017. DOI: 10.1109/ICRS.2017.8272853.
- [50] K. S. Morgan, D. L. McMurtrey, B. H. Pratt y M. J. Wirthlin, “A Comparison of TMR With Alternative Fault-Tolerant Design Techniques for FPGAs,” *IEEE Transactions on Nuclear Science*, 2007. DOI: 10.1109/TNS.2007.910871.
- [51] J. Anwer, M. Platzner y S. Meisner, “FPGA Redundancy Configurations: An Automated Design Space Exploration,” *IEEE International Parallel & Distributed Processing Symposium Workshops*, 2014. DOI: 10.1109/IPDPSW.2014.37.
- [52] “Triple module redundancy design techniques for Virtex FPGAs,” *Xilinx*, 2001.

- [53] F. Kastensmidt, L. Sterpone, L. Carro y M. Reorda, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," *Design, Automation and Test in Europe*, 2005. DOI: 10.1109/DATe.2005.229.
- [54] V. S. Kharchenko, V. S. Kharchenko, A. A. Siora, V. V. Sklyar y V. I. Tokarev, "Diversity-oriented FPGA-based NPP I&C systems: safety assesment, development, implementation," *International Conference on Nuclear Engineering*, 2010. DOI: 10.1115/ICONE18-29754.
- [55] V. Kharchenko, V. Sklyar y A. V. Volkoviy, "Defence-in-Depth and Diversity Analysis of FPGA-Based NPP I&C Systems: Conception, Technique and Tool," *International Conference on Nuclear Engineering and the ASME*, 2012. DOI: 10.1115/ICONE20-POWER2012-54349.
- [56] R. A. Ghignone et al., "Modelling, Simulation and Code Generation for Electronic Railway Interlocking Systems," *IEEE Latin America Transactions*, 2021. DOI: 10.1109/TLA.2021.9423859.
- [57] M. Banci y A. Fantechi, "Geographical Versus Functional Modelling by Statecharts of Interlocking Systems," *Electronic Notes in Theoretical Computer Science*, 2005. DOI: 10.1016/j.entcs.2004.08.055.
- [58] M. S. Durmus, S. Takai y M. T. Soylemez, "Decision-Making Strategies in Fixed-Block Railway Signaling Systems: A Discrete Event Systems Approach," *EEJ Transactions on Electrical and Electronic Engineering*, 2015. DOI: 10.1002/tee.22052.
- [59] W. Wong, "A Simple Graph Theory and Its Application in Railway Signalling," *International Workshop on the HOL Theorem Proving System and Its Applications*, 2002. DOI: 10.1109/HOL.1991.596304.
- [60] D. Wang, X. Chen y H. Huang, "A graph theory based new approach to route location in railway interlocking," *Computers & Industrial Engineering*, 2013. DOI: 10.1016/j.cie.2013.09.019.
- [61] I. Bethke, "Selecting sets of disjoint paths in a railway graph," *Graduate project*, 2005.
- [62] M. A. N. Oz, I. Sener, O. T. Kaymakci, I. Ustoğlu y G. Cansever, "Topology Based Automatic Formal Model Generation for Point Automation Systems," *Informartion technology and control*, 2015. DOI: 10.5755/j01.itc.44.1.7382.
- [63] T. Ye, X. Zhang, Y. Zhang y J. Liu, "Towards a Railway Topology Ontology toIntegrate and Query Rail Data Silos," *International Semantic Web Conference*,
- [64] E. Roanes-Lozano y L. M. Laita, "An applicable topology independent model for railway interlocking systems," *Mathematics and Computers in Simulation (MATCOM)*, 1998. DOI: 10.1016/S0378-4754(97)00093-1.
- [65] A. Kuzu, O. Songuler, A. Sonat, S. Turk, B. Birol y E. H. Dogruguvan, "Automatic interlocking table generation from railway topology," *IEEE International Conference on Mechatronics*, 2011. DOI: 10.1109/ICMECH.2011.5971206.
- [66] U. R. S. Department, "Towards a Universal Topology Model for Railways and Data Exchange Format for Infrastructure," *UIC RailTopoModel and railML Conference*, 2015.
- [67] "IEC 61508-1-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements," *International Standard*, 2010.
- [68] "IEC 61508-2-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems," *International Standard*, 2010.
- [69] "IEC 61508-3-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements," *International Standard*, 2010.

- [70] “IEC 61508-4-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 4: Definitions and abbreviations,” *International Standard*, 2010.
- [71] “IEC 61508-5-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 5: Examples of methods for the determination of safety integrity levels,” *International Standard*, 2010.
- [72] “IEC 61508-6-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3,” *International Standard*, 2010.
- [73] “IEC 61508-7-2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 7: Overview of techniques and measures,” *International Standard*, 2010.
- [74] “Developing Safety Critical Applications that Meet IEC 61508 Standards, Using System-on-Chip Devices with Embedded ARM Cortex-M3 and FPGA,” *Microsemi*, 2013.
- [75] E. Gracic, A. Hayek y J. Börcsök, “Implementation of a Fault-Tolerant System Using Safety-Related Xilinx Tools Conforming to the Standard IEC 61508,” *International Conference on System Reliability and Science (ICSR)*, 2016. doi: 10.1109/ICSR.2016.7815842.
- [76] J. C. Knight, “Safety Critical Systems: Challenges and Directions,” *International Conference on Software Engineering (ICSE)*, 2002.
- [77] T. Mens, J. Magee y B. Rumpe, “Evolving software architecture descriptions of critical systems,” *IEEE Computer Society*, 2010.
- [78] G. Chen, D. Fan y J. Yang, “Research and Implementation of Fault-Tolerant Computer Interlocking System,” *International Conference on Computational Intelligence and Software Engineering*, 2010. doi: 10.1109/CISE.2010.5676767.
- [79] CENELEC, “EN 50126-1 Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 1: Generic RAMS Process,” 2018.
- [80] CENELEC, “EN 50126-2 Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 2: Systems Approach to Safety,” 2018.
- [81] CENELEC, “EN 50126-3 Railway applications. The specification and demonstration of reliability, availability, maintainability and safety (RAMS) Guide to the application of EN 50126-1 for rolling stock RAM,” 2008.
- [82] CENELEC, “EN 50126-4 Railway applications. The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS). Part 4. Functional Safety. Electrical/Electronic/Programmable electronic systems,” 2012.
- [83] CENELEC, “EN 50126-5 Railway applications. The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS). Part 5. Functional Safety. Software,” 2012.
- [84] CENELEC, “EN 50128 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems,” 2012.
- [85] CENELEC, “EN 50129 Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling,” 2020.
- [86] “Reliability, Safety and Security of railway systems,” *First International Conference, RSS-Rail 2022*, 2016. doi: 10.1007/978-3-031-05814-1.
- [87] M. Bellek, “Design and RAMS analysis of railway interlocking systems using formal methods,” *Department of Electrical Engineering, Istanbul Technical University*, 2013.

- [88] Q. Mahboob y E. Zio, "Handbook of RAMS in railway systems theory and practice," *CRC Press*, 2018.
- [89] T. Kishi y N. Noda, "Software Design based on Architecture Conformance," 1999.
- [90] M. S. Durmuş, U. Yıldırım, O. Eriş y M. T. Söylemez, "Safety-Critical Interlocking Software Development Process for Fixed-Block Signalization Systems," *Symposium on Control in Transportation Systems*, 2012. DOI: 10.3182/20120912-3-BG-2031.00034.
- [91] "The Xilinx Isolation Design Flow for Fault-Tolerant Systems," *Xilinx*, 2013.
- [92] Y. Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga y T. Sueyoshi, "Improving the Robustness of a Softcore Processor against SEUs by using TMR and Partial Reconfiguration," *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 2010. DOI: 10.1109/FCCM.2010.16.
- [93] T. Tamandl y P. Preininger, "Online Self Tests for Microcontrollers in Safety Related Systems," *IEEE International Conference on Industrial Informatics*, 2007. DOI: 10.1109/INDIN.2007.4384745.
- [94] E.-j. Joung y K.-h. Shin, "Suggestions of Development Methodology for Railway Software," *International Technical Conference on Circuits/Systems, Computers and Communications*, 2008.
- [95] U. Yıldırım, M. S. Durmuş y M. T. Söylemez, "Application of Functional Safety on Railways Part II: Software Development," *Asian Control Conference (ASCC)*, 2011.
- [96] R. Olay, "Safety Critical Applications," *Microsemi Industrial Business Manager*, 2012.
- [97] "Development of Safety Related Systems," *Lattice Semiconductor*, 2015.
- [98] "Functional safety in process instrumentation with SIL rating," *Siemens*, 2003.
- [99] D. Streitferdt, A. Zimmermann, J. Schaffner y M. Kallenbach, "Component-Wise Software Certification for Safety-Critical Embedded Devices," *nnual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, 2017. DOI: 10.1109/IEMECON.2017.8079584.
- [100] A. Drozd, V. Kharchenko, A. Siora y V. Sklyar, "Component-Based Safety-Oriented On-Line Testing of Digital Systems," *Component-Based Safety-Oriented On-Line Testing of Digital Systems*, 2010. DOI: 10.1109/EWDTs.2010.5742122.
- [101] J. Drozd, A. Drozd, S. Antoshchuk y V. Kharchenko, "Natural development of the resources in design and testing of the computer systems and their components," *International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, 2013. DOI: 10.1109/IDAACS.2013.6662656.
- [102] A. Drozd, V. Kharchenko, S. Antoshchuk, J. Sulima y M. Drozd, "Checkability of the Digital Components in Safety-Critical Systems: Problems and Solutions," *East-West Design & Test Symposium (EWDTs)*, 2011. DOI: 10.1109/EWDTs.2011.6116606.
- [103] A. Drozd, V. Kharchenko, S. Antoshchuk, J. Drozd, M. Lobachev y J. Sulima, "The Use of Natural Resources for Increasing a Checkability of the Digital Components in Safety-Critical Systems," *East-West Design & Test Symposium (EWDTs)*, 2013. DOI: 10.1109/EWDTs.2013.6673209.
- [104] J. Drozd, O. Drozd, S. Antoshchuk, A. Kushnerov y V. Nikul, "Effectiveness of Matrix and Pipeline FPGA-Based Arithmetic Components of Safety-Related Systems," *International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 2015. DOI: 10.1109/IDAACS.2015.7341410.

- [105] B.-g. Cai1, C.-m. Jin, L.-c. Ma, Y. Cao y H. Nakamura, “Analysis on the application of on-chip redundancy in the safety-critical system,” *EICE Electronics Express*, 2014. DOI: 10.1587/elex.11.20140153.
- [106] R. Girardey, M. Hübner y J. Becker, “Safety Aware Place and Route for On-Chip Redundancy in Safety Critical Applications,” *IEEE Computer Society Annual Symposium on VLSI*, 2010. DOI: 10.1109/ISVLSI.2010.51.
- [107] M. S. Durmuş, O. Eriş, U. Yıldırım y M. T. Söylemez, “A New Voting Strategy in Diverse Programming for Railway Interlocking Systems,” *International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, 2011. DOI: 10.1109/TMEE.2011.6199304.
- [108] F. Mazzanti, A. Ferrari y G. O. Spagnolo, “Towards formal methods diversity in railways: an experience report with seven frameworks,” *International Journal on Software Tools for Technology Transfer*, 2018. DOI: 10.1007/s10009-018-0488-3.
- [109] I. Zaza, “Report of a comparative analysis of the Interlocking Systems,” *Department of Information Engineering (DINFO), University of Florence*, 2013.
- [110] K. Shimamura et al., “A Fail-safe Microprocessor Using Dual Synthesizable Processor Cores,” *AP-ASIC'99. First IEEE Asia Pacific Conference on ASICs (Cat. No.99EX360)*, 1999. DOI: 10.1109/APASIC.1999.824026.
- [111] V. Kharchenko, E. Babeshko, V. Sklyar, A. Siora y V. Tokarev, “Combined Implementation of Dependability Analysis Techniques for NPP I&C Systems Assessment,” *Energy and Power Engineering*, 2011.
- [112] G. Park, Y. Yu, H. T. Kim, Y. Kwon, H. Park y C. H. Jeong, “Regulatory Issues on Using Programmable Logic Device in Nuclear Power Plants,” *Korean Nuclear Society*, 2012.
- [113] M. Kumar, A. Kabra, G. Karmakar y P. Marathe, “A Review of Defences against Common Cause Failures in Reactor Protection Systems,” *International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*, 2015. DOI: 10.1109/ICRITO.2015.7359232.
- [114] “Trackguard Westrace Mk II,” Siemens Mobility. (2018), dirección: <https://assets.new.siemens.com/siemens/assets/api/uuid:688f3a74-6f27-446d-81b7-899827b02631/trackguard-westrace-mkii-en.pdf> (visitado 11-02-2024).
- [115] “Alstoms Smartlock equipment at the heart of modular signalling project in the UK,” Alstom Transport. (2011), dirección: <https://www.alstom.com/press-releases-news/2011/3/Alstom-Smartlock> (visitado 11-02-2024).
- [116] “The Hitachi WSP2G Computer Based Interlocking System,” Hitachi Rail. (2022), dirección: <https://www.rissb.com.au/wp-content/uploads/2022/07/FastTrack-NNewsletter-July-2022-Maharajahs-Express.pdf> (visitado 11-02-2024).
- [117] Thales. “Route Control Systems.” (2024), dirección: <https://www.thalesgroup.com/en/route-control-systems> (visitado 11-02-2024).
- [118] Bombardier. “Bombardier Marks a Century of Rail Control Leadership.” (2015), dirección: <https://bombardier.com/en/media/news/bombardier-marks-century-rail-control-leadership> (visitado 11-02-2024).
- [119] Kyosan. “Railway Signaling Solutions, Interlocking Equipment.” (2024), dirección: <https://www.kyosan.co.jp/english/product/signal03.html> (visitado 11-02-2024).
- [120] Hima. “COTS Rail Applications – a global movement that is growing rapidly.” (2024), dirección: <https://www.hima.com/en/industries-solutions/cots-rail-references> (visitado 11-02-2024).

- [121] CRRC. “CRRC SRI won the signal system order for China’s first hydrogen energy tram,” CRRC Qingdao Sifang Rolling Stock Research Institute. (2018), dirección: <https://www.crrcgc.cc/sfsen/g11483/s21363/t291490.aspx> (visitado 11-02-2024).
- [122] CAF. “Quasar Q4, Electronic Interlocking,” CAF Signalling. (2024), dirección: <https://www.cafsignalling.com/en/products-and-services/electronic-interlocking/> (visitado 11-02-2024).
- [123] Transmashholding. “Innovative Rail Traffic Control Systems.” (2024), dirección: <https://tmhsmart.ru/en/> (visitado 11-02-2024).
- [124] “Definition and technology trend of train signaling system,” Hyundai Rotem. (2024), dirección: <https://tech.hyundai-rotem.com/en/new-normal/definition-and-technology-trend-of-train-signaling-system/> (visitado 11-02-2024).
- [125] “GE Transportation Increases Safety and Efficiency on TriMet Transit Network in Portland,” General Electric transportation. (2010), dirección: <https://www.ge.com/news/taxonomy/term/2344> (visitado 11-02-2024).
- [126] Caterpillar. “Signalling Products & Systems.” (2024), dirección: <https://www.progressrail.com/en/Segments/Infrastructure/Signaling/ECM.html> (visitado 11-02-2024).
- [127] “Electronic interlocking systems with remote control,” Stadler Rail. (2024), dirección: <https://www.stadlerrail.com/en/signalling/branchline/> (visitado 11-02-2024).
- [128] “IRS 30100 -RailTopoModel - Railway infrastructure topological model,” *International Union of Railways*, 2016.
- [129] P. Nesi, I. Zaza y P. Bellini, “Raiss-Railway Signalling : Safety and Security,” *Department of Information Engineering (DINFO), University of Florence*, 2014.
- [130] “Feasibility study RailTopoModel,” *International Union of Railways*, 2013.
- [131] A. Hlubuček, “RailTopoModel and RailML3 in overall context,” *Acta Polytechnica CTU Proceedings*, 2017. DOI: 10.14311/APP.2017.11.0016.
- [132] M. Bosschaart, E. Quaglietta, B. Janssen y R. M. P. Goverde, “Efficient formalization of railway interlocking data in RailML,” *Information Systems*, 2015. DOI: 10.1016/j.is.2014.11.007.
- [133] “Using railML for Exchanging Timetable Data,” 2017.
- [134] Railml.org. “Home - railML.org.” (2022), dirección: <http://www.railml.org> (visitado 14-02-2024).
- [135] A. Nash, D. Huerlimann, J. Schuette y V. P. Krauss, “RailML – A standard data interface for railroad applications,” *Comprail*, 2004.
- [136] Railml.org. “RailML 3.2 Common.” (2022), dirección: <http://wiki3.railml.org/DataModel/3.2/CO/> (visitado 14-02-2024).
- [137] Railml.org. “RailML 3.2 Infrastructure.” (2022), dirección: <http://wiki3.railml.org/DataModel/3.2/IS/> (visitado 14-02-2024).
- [138] Railml.org. “RailML 3.2 Interlocking.” (2022), dirección: <http://wiki3.railml.org/DataModel/3.2/IL/> (visitado 14-02-2024).
- [139] Railml.org. “RailML 3.2 RollingStock.” (2022), dirección: <http://wiki3.railml.org/DataModel/3.2/RS/> (visitado 14-02-2024).
- [140] Railml.org. “RailML 3.2 TimeTable.” (2022), dirección: <http://wiki3.railml.org/DataModel/3.2/TT/> (visitado 14-02-2024).
- [141] Railml.org. “railML® partners.” (2024), dirección: <https://www.railml.org/en/introduction/partners.html> (visitado 14-02-2024).

- [142] Maprex. “Maprex Train Simulator.” (2024), dirección: <https://maprex.co.kr/> (visitado 14-02-2024).
- [143] “IVU.rail,” IVU Traffic Technologies. (2024), dirección: <https://www.ivu.com/solutions/highlights/ivurail> (visitado 14-02-2024).
- [144] Railml.org. “railVIVID: The railML Viewer & Validator.” (2024), dirección: <https://www.railml.org/en/user/railvivid.html> (visitado 14-02-2024).
- [145] S.-S. John. “VIS-ALL 3D.” (2024), dirección: <https://john-software.de/produkte/vis-all-3d/> (visitado 14-02-2024).
- [146] “Railway Infrastructure and Layout Aided Designer,” Design 4 Rail y Track Planner. (2024), dirección: <https://design4rail.com/service/d4rhorizon/> (visitado 14-02-2024).
- [147] Wikipedia. “Universal Serial Bus (USB).” (2024), dirección: <https://en.wikipedia.org/wiki/USBs> (visitado 18-07-2024).
- [148] Wikipedia. “Bluetooth.” (2024), dirección: <https://en.wikipedia.org/wiki/Bluetooth> (visitado 18-07-2024).
- [149] Wikipedia. “General Packet Radio Service (GPRS).” (2024), dirección: https://en.wikipedia.org/wiki/General_Packet_Radio_Service (visitado 18-07-2024).
- [150] “Global Rolling Stock Market: Industry Analysis and Forecast (2023-2029),” Maximize Market Research. (2024), dirección: <https://www.maximizemarketresearch.com/market-report/global-rolling-stock-market/26747/> (visitado 11-02-2024).
- [151] U. Yıldırım, M. S. Durmus y M. T. Soylemez, “Automatic Interlocking Table Generation for Railway Stations using Symbolic Algebra,” *IFAC Symposium on Control in Transportation Systems*, 2012. DOI: 10.3182/20120912-3-BG-2031.00035.
- [152] M. S. Durmus, “A control and automation engineering approach to railway interlocking system design,” 2014.
- [153] I. Ristic, “Automated development of railway signalling control tables: A case study from Serbia,” *Mechatronics and Intelligent Transportation Systems*, 2023. DOI: 10.56578/mits020404.
- [154] A. E. Haxthausen, “Automated generation of formal safety conditions from railway interlocking tables,” *International Journal on Software Tools for Technology Transfer*, 2013. DOI: 10.1007/s10009-013-0295-9.
- [155] Q. Qu, “Research on Life Evaluation of Computer Interlocking System Based on Neural Network,” *International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, 2018. DOI: 10.1109/SDPC.2018.8664937.
- [156] M. T. Söylemez, M. S. Durmuş, U. Yıldırım, S. Türk y A. Sonat, “The Application of Automation Theory to Railway Signalization Systems: The Case of Turkish National Railway Signalization Project,” *IFAC World Congress*, 2011. DOI: 10.3182/20110828-6-IT-1002.03755.
- [157] X. Chen, Y. He y H. Huang, “An approach to automatic development of interlocking logic based on statechart,” *Enterprise Information Systems*, 2011. DOI: 10.1080/17517575.2011.575475.
- [158] M. F. Ariyachandra e I. Brilakis, “Application of Railway Topology for the Automated Generation of Geometric Digital Twins of Railway Masts,” *European Conference on Product & Process Modelling*, 2021. DOI: 10.1201/9781003191476-52.
- [159] C. Xiangxian, H. Hai y H. Yulin, “Automatic Generation of Relay Logic for Interlocking System Based on Statechart,” *Second World Congress on Software Engineering*, 2010. DOI: 10.1109/WCSE.2010.31.

- [160] A. Das, M. K. Gangwar, D. Ghosh, C. Mandal, A. Sengupta y M. M. Waris, “Automatic Generation of Route Control Chart From Validated Signal Interlocking Plan,” *IEEE Transactions on intelligent transportation systems*, 2020. DOI: 10.1109/TITS.2020.2993794.
- [161] A. Rehman, S. Latif y N. A. Zafar, “Automata Based Railway Gate Control System at Level Crossing,” *International Conference on Communication Technologies (ComTech)*, 2019. DOI: 10.1109/COMTECH.2019.8737833.
- [162] M. A. N. Oz, I. Sener, O. T. Kaymakci, I. Ustoglu y G. Cansever, “A Tool for Automatic Formal Modeling of Railway Interlocking Systems,” *International Conference on Computer as a Tool (EUROCON)*, 2015. DOI: 10.1109/EUROCON.2015.7313752.
- [163] T. Gonschorek, L. Bedau y F. Ortmeier, “Automatic Model-based Verification of Railway Interlocking Systems using Model Checking,” *Proceedings of ESREL*, 2018.
- [164] L. Zhang, “Railway Signal Interlocking Logic Simulation System,” *International Conference on Control and Computer Vision*, 2019. DOI: 10.1145/3341016.3341017.
- [165] A. Ferrari, A. Fantechi, G. Magnani, D. Grasso y M. Tempestini, “The Metro Rio case study,” *Science of Computer Programming*, 2013. DOI: 10.1016/j.scico.2012.04.003.
- [166] M. Banci, A. Fantechi y S. Gnesi, “Some experiences on Formal specification of Railway Interlocking Systems using Statecharts,” 2005.
- [167] K. Winter y N. J. Robinson, “Modelling Large Railway Interlockings and Model Checking Small Ones,” *Australasian computer science conference*, 2003.
- [168] P. James, “SAT-based Model Checking and its applications to Train Control Systems,” *Department of Computer Science, Swansea University*, 2010.
- [169] B. T. Celebi y O. T. Kaymakci, “Verifying the accuracy of interlocking tables for railway signalling systems using abstract state machines,” *Journal of Modern Transportation*, 2016. DOI: 10.1007/s40534-016-0119-1.
- [170] C. George y A. Haxthausen, “The logic of the RAISE specification language,” *Computing and Informatics*, 2003. DOI: 10.1007/978-3-540-74107-7_7.
- [171] S. Ghosh, A. Das, N. Basak, P. Dasgupta y A. Katiyar, “Formal Methods for Validation and Test Point Prioritization in Railway Signaling Logic,” *IEEE Transactions on Intelligent Transportation Systems*, 2017. DOI: 10.1109/TITS.2016.2586512.
- [172] S. Chadwick, P. James, M. Roggenbach y T. Wetner, “Formal Methods for Industrial Interlocking Verification,” *International Conference on Intelligent Rail Transportation (ICIRT)*, 2018. DOI: 10.1109/ICIRT.2018.8641579.
- [173] Q. Cappart, C. Limbrée, P. Schaus, J. Quilbeuf, L.-M. Traonouez y A. Legay, “Verification of Interlocking Systems Using Statistical Model Checking,” *International Symposium on High Assurance Systems Engineering (HASE)*, 2017. DOI: 10.1109/HASE.2017.10.
- [174] J. Mocki y L. Vlacic, “Railway interlocking process - Building a base for formal methods,” *IEEE International Conference on Intelligent Rail Transportation Proceedings*, 2013. DOI: 10.1109/ICIRT.2013.6696284.
- [175] S. Ghosh, A. Das, N. Basak, P. Dasgupta y A. Katiyar, “Formal Methods for Validation and Test Point Prioritization in Railway Signaling Logic,” *IEEE Transactions on intelligent transportation systems*, 2016. DOI: 10.1109/TITS.2016.2586512.
- [176] A. Hernando, R. Maestre y E. Roanes-Lozano, “A New Algebraic Approach to Decision Making in a RailwayInterlocking System Based on Preprocess,” *Mathematical Problems in Engineering*, 2018. DOI: 10.1155/2018/4982974.

- [177] J. Falampin, H. Le-Dang, M. Leuschel y D. Plagge, “Improving Railway Data Validation with ProB,” *Industrial Deployment of System Engineering Methods*, 2013. doi: 10.1007/978-3-642-33170-1_4.
- [178] A. Iliasov, P. Stankaitis y D. E. Adjepon-Yamoah, “Static Verification of Railway Schema and Interlocking Design Data,” *Proceedings of Reliability, Safety, and Security of Railway Systems (RSSRail)*, 2016. doi: 10.1007/978-3-319-33951-1_9.
- [179] A. Ferrari y M. ter Beek, “Formal Methods in Railways: A Systematic Mapping Study,” *ACM Computing Surveys*, 2022. doi: 10.1145/3520480.
- [180] B. Luteberget y C. Johansen, “Efficient verification of railway infrastructure designs against standard regulations,” 2017. doi: 10.1007/s10703-017-0281-z.
- [181] G. Lukács y T. Bartha, “Formal Modeling and Verification of the Functionality of Electronic Urban Railway Control Systems Through a Case Study,” *Urban Rail Transit*, 2022. doi: 10.1007/s40864-022-00177-8.
- [182] D. Basile, F. Mazzanti y A. Ferrari, “Experimenting with Formal Verification and Model-Based Development in Railways: The Case of UMC and Sparx Enterprise Architect,” *Formal Methods for Industrial Critical Systems*, 2023. doi: 10.1007/978-3-031-43681-9_1.
- [183] A. Ferrari, A. Fantechi, S. Gnesi y G. Magnani, “Model-Based Development and Formal Methods in the Railway Industry,” *IEEE Software*, 2013. doi: 10.1109/MS.2013.44.
- [184] S. Ghosh, A. Das, N. Basak, P. Dasgupta y A. Katiyar, “Formal Methods for Validation and Test Point Prioritization in Railway Signaling Logic,” *IEEE Transactions on Intelligent Transportation Systems*, 2016. doi: 10.1109/TITS.2016.2586512.
- [185] A. Iliasov y A. Romanovsky, “Formal Analysis of Railway Signalling Data,” *International Symposium on High Assurance Systems Engineering (HASE)*, 2016. doi: 10.1109/HASE.2016.44.
- [186] A. Iliasov y A. Romanovsky, “SafeCap Domain Language for Reasoning about Safety and Capacity,” *Workshop on Dependable Transportation Systems/Recent Advances in Software Dependability*, 2012. doi: 10.1109/WDTS-RASD.2012.11.
- [187] A. Ferrari et al., “Survey on Formal Methods and Tools in Railways: The ASTRail Approach,” 2019. doi: 10.1007/978-3-030-18744-6_15.
- [188] A. Iliasov, D. Taylor, L. Laibinis y A. Romanovsky, “Formal Verification of Signalling Programs with SafeCap,” *International Conference on Computer Safety, Reliability and Security*, 2018. doi: 10.1007/978-3-319-99130-6_7.
- [189] M. Huber y S. King, “Towards an Integrated Model Checker for Railway Signalling Data,” *International Symposium of Formal Methods Europe*, 2002.
- [190] S. Busard, Q. Cappart, C. Limbree, C. Pecheur y P. Schaus, “Verification of railway interlocking systems,” *International Workshop on Engineering Safety and Security Systems*, 2015. doi: 10.4204/EPTCS.184.2.
- [191] A. Iliasov, I. Lopatkin y A. Romanovsky, “The SafeCap Platform for Modelling Railway Safety and Capacity,” *International Conference on Computer Safety, Reliability and Security*, 2013. doi: 10.1007/978-3-642-40793-2_12.
- [192] A. Iliasov, I. Lopatkin y A. Romanovsky, “Practical Formal Methods in Railways – The SafeCap Approach,” *Proceedings of Reliable Software Technologies (Ada-Europe)*, 2014. doi: 10.1007/978-3-319-08311-7_14.
- [193] P. Behm, P. Benoit, A. Faivre y J.-M. Meynadier, “Meteor A Successful Application of B in a Large Project,” *International Symposium on Formal Methods*, 1999.

- [194] F. Badeau y A. Amelot, “Using B as a High Level Programming Language in an Industrial Project: Roissy VAL,” 2005. doi: 10.1007/11415787_20.
- [195] M. Butler et al., “The First Twenty-Five Years of Industrial Use of the B-Method,” *Formal Methods for Industrial Critical Systems*, 2020. doi: 10.1007/978-3-030-58298-2_8.
- [196] J.-R. Abrial, “Formal Methods: Theory Becoming Practice,” *Universal Computer Science*, 2007.
- [197] M. N. Menéndez. “Repositorio bibliográfico.” (2024), dirección: <https://github.com/Martin-N-Menendez/PhDThesis/tree/main/Bibliografia> (visitado 14-02-2024).
- [198] A. Ferrari, F. Mazzanti, D. Basile y M. ter Beek and, “Systematic Evaluation and Usability Analysis of Formal Tools for Railway System Design,” *IEEE Transactions On Software Engineering*, 2021. doi: 10.1109/TSE.2021.3124677.
- [199] A. Ferrari, F. Mazzanti, D. Basile, M. H. ter Beek y A. Fantechi, “Comparing Formal Tools for System Design: a Judgment Study,” *IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2020.
- [200] CONICET-GICSAFE. “Grupo de Investigación en Calidad y Seguridad de las Aplicaciones Ferroviarias (GICSAFE).” (2023), dirección: <https://sites.google.com/view/conicet-gicsafe/inicio> (visitado 10-06-2023).
- [201] “Trenes Argentinos,” Trenes Argentinos. (2024), dirección: <https://www.argentina.gob.ar/transporte/trenes> (visitado 11-02-2024).
- [202] CONICET. “Primer premio INNOVAR 2018 para un Monitor de Barreras para optimizar el sistema ferroviario.” (2018), dirección: <https://www.conicet.gov.ar/primer-premio-innovar-2018-para-un-monitor-de-barreras-para-optimizar-el-sistema-ferroviario/> (visitado 10-06-2023).
- [203] Argentina.gob.ar. “Se conocieron los ganadores de INNOVAR 2018.” (2018), dirección: <https://www.argentina.gob.ar/noticias/se-conocieron-los-ganadores-de-innovar-2018> (visitado 10-06-2023).
- [204] M. Menéndez, S. Germino, F. Larosa y A. Lutenberg, “Automatic generation of VHDL code for a railway interlocking system,” *International Journal of Embedded Systems*, 2022. doi: 10.1504/IJES.2021.121088.
- [205] L. Dórdolo et al., “Modular system for the monitoring and management of railway signaling equipment,” *IEEE Latin America Transactions*, 2020. doi: 10.1109/TITS.2023.3317256.
- [206] M. N. Menéndez, S. Germino, L. D. Díaz-Charris y A. Lutenberg, “Automatic Railway Signaling Generation for Railways Systems Described on Railway Markup Language (railML),” *IEEE Transactions on Intelligent Transportation Systems*, 2023. doi: 10.1109/TITS.2023.3317256.
- [207] S. Germino, M. Menéndez y A. Lutenberg, “An XSLT-Based Proposal to Ease Embedded Critical Systems Tools Implementation, Verification, Validation, Testing, and Certification Efforts,” *IEEE embedded systems letters*, 2022. doi: 10.1109/LES.2022.3221810.
- [208] A. Avizienis, J.-C. Laprie, B. Randell y C. Landwehr, “Basic Concepts and Taxonomy of Dependable and Secure Computing,” *IEEE Transactions on Dependable and Secure Computing*, 2004. doi: 10.1109/TDSC.2004.2.
- [209] M. Drozd y A. Drozd, “Safety-Related Instrumentation and Control Systems and a Problem of the Hidden Faults,” *International Conference on Digital Technologies (DT)*, 2014. doi: 10.1109/DT.2014.6868692.
- [210] J.-F. Demoutiez, “The foundation for an universal Infraestructure Data Exchange Format,” *ERIM Conference*, 2013.

- [211] N. A. Zafar, S. A. Khan y K. Araki, "Towards the safety properties of moving block railway interlocking system," *International Journal of Innovative Computing, Information and Control*, 2011.
- [212] Durmuş, M. S., Takai, S., Söylemez y M. T., "Fault Diagnosis in Fixed-Block Railway Signaling Systems: A DiscreteEvent Systems Approach," *EEJ Transactions on electrical and electronic engineering*, 2014. DOI: 10.1002/tee.22001.
- [213] J. Široký, P. Nachtigall, E. Tischer y J. Gašparík, "Simulation of Railway Lines with a Simplified Interlocking System," *MDPI*, 2021. DOI: 10.3390/su13031394.
- [214] C. Zheng, S. Assaf y B. Eynard, "Towards modelling and standardisation techniques for railway infrastructure," *Product Lifecycle Management and the Industry of the Future*, 2018. DOI: 10.1007/978-3-319-72905-3_23.
- [215] A. Hernando, E. Roanes-Lozano, J. L. Galan-García y G. Aguilera-Venegas, "Decision making in railway interlocking systems based on calculating the remainder of dividing a polynomial by a set of polynomials," *Electronic Research Archive*, 2023. DOI: 10.3934/era.2023313.
- [216] M. Aanes y H. P. Thai, "Modelling and Verification of Relay Interlocking Systems," 2012.
- [217] G. Pinter, Z. Micskei e I. Majzik, "Supporting design and development of safety critical applications by model based tools," 2009.
- [218] Z. Wang, C.-g. Geng1, X.-x. Chen, D. Wang, H. Huang y A.-a. Guan, "Modeling Safety-Critical System Requirements with Hierarchical State Machine," *International Conference on Information Science, Electronics and Electrical Engineering*, 2014. DOI: 10.1109/InfoSEEE.2014.6947759.
- [219] S. Bacherini et al., "Modelling a Railway Signalling System using SDL," 2003. dirección: <https://api.semanticscholar.org/CorpusID:17261702>.
- [220] "Model Driven Engineering of railway control systems with the openETCS process," 2014.
- [221] B. Aristyo, K. Pradityo, T. Tamba, Y. Nazaruddin y A. Widjyotriatmo, "Model Checking-based Safety Verification of a Petri Net Representation of Train Interlocking Systems," *Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2018. DOI: 10.23919/SICE.2018.8492661.
- [222] U. Yildirim, M. S. Durmus y M. T. Söylemez, "Fail-Safe Signalization and Interlocking Design for a Railway Yard: An Automation Petri Net Approach," *International Symposium on Intelligent and Manufacturing Systems*, 2010.
- [223] B. Malakar y B. Roy, "Railway Fail-Safe Signalization and Interlocking Design Based On Automation Petri Net," *International Conference on Information Communication and Embedded Systems*, 2014. DOI: 10.1109/ICICES.2014.7034154.
- [224] C.-K. Chen, "A Petri net design of FPGA-based controller for a class of nuclear I&C systems," *Institute of Nuclear Energy Research*, 2011. DOI: 10.1016/j.nucengdes.2011.04.004.
- [225] X. Hei, L. Chang, W. Ma, J. Gao y G. Xie, "Automatic transformation from UML Statechart to Petri Nets for safety analysis and verification," *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, 2011. DOI: 10.1109/ICQR2MSE.2011.5976760.
- [226] J. Liu, Y. Zhang, J. Han, J. He, J. Sun y T. Zhou, "Intelligent Hazard-Risk Prediction Modelfor Train Control Systems," *IEEE Transactions on intelligent transportation systems*, 2020. DOI: 10.1109/TITS.2019.2945333.

- [227] G. Liu y Z. Hou, "Railway Traffic Object Detection Using DifferentialFeature Fusion Convolution Neural Network," *IEEE Transactions on intelligent transportation systems*, 2020. DOI: 10.1109/TITS.2020.2969993.
- [228] "Adaptive Iterative Learning Control for SubwayTrains Using Multiple-Point-Mass DynamicModel Under Speed Constraint," *IEEE Transactions on intelligent transportation systems*, 2020. DOI: 10.1109/TITS.2020.2970000.
- [229] H. Su, M. Chai, H. Liu, J. Chai y C. Yue, "A Model-Based Testing System for Safety of Railway Interlocking," *International Conference on Intelligent Transportation Systems (ITSC)*, 2022. DOI: 10.1109/ITSC55140.2022.9922557.
- [230] A. Lewiński y K. Trzaska–Rycaj, "The Safety Related Software for Railway Control with Respect to Automatic Level Crossing Signaling System," *International Conference on Transport Systems Telematics*, 2010.
- [231] H. Wang, C. Gao y S. Liu, "Model-based Software Development for Automatic Train Protection System," *Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA)*, 2009. DOI: 10.1109/PACIIA.2009.5406387.
- [232] N. Silva y R. Lopes, "Practical Experiences with real-world systems: Security in the World of Reliable and Safe Systems," *Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, 2013. DOI: 10.1109/DSNW.2013.6615515.
- [233] "Aplicacion de normativa CENELEC a desarrollo y explotacion de material rodante," *CAF*,
- [234] H. Jonsson, S. Larsson y S. Punnekkat, "Agile Practices in Regulated Railway Software Development," *International Symposium on Software Reliability Engineering Workshops*, 2012. DOI: 10.1109/ISSREW.2012.80.
- [235] A. Ceccarelli, I. Majzik, D. Iovino, F. Caneschi, G. Pinter y A. Bondavalli, "A Resilient SIL 2 Driver Machine Interface for Train Control Systems," *International Conference on Dependability of Computer Systems DepCoS-RELCOMEX*, 2008. DOI: 10.1109/DepCoS-RELCOMEX.2008.33.
- [236] A. Andrashov, V. Kharchenko, V. Sklyar, A. Siora y L. Reva, "Verification of FPGA-based NPP I&C systems: general approach and techniques," *International Conference on Nuclear Engineering*, 2011.
- [237] G. Y. Park, C. H. Jung y D. I. Kim, "A Study on the Verification and Validation of Programmable Logic Component in A Nuclear Power Plant," *Korean Nuclear Society*, 2007.
- [238] A. Idani, Y. Ledru y G. Vega, "Alliance of model-driven engineering with a proof-based formal," *Innovations in Systems and Software Engineering*, 2020. DOI: 10.1007/s11334-020-00366-3.
- [239] A. E. Haxthausen, A. Fantechi y G. Gori, "Decomposing the Verification of Interlocking Systems," *Lecture Notes in Computer Science*, 2023. DOI: 10.1007/978-3-031-40132-9_7.
- [240] P. James, F. Moller y M. Roggenbach, "Software Model Checking of Interlocking Programs," *Lecture Notes in Computer Science*, 2023. DOI: 10.1007/978-3-031-40132-9_9.
- [241] M. N. Menéndez. "Layouts analyzed with RNA." (2022), dirección: <https://github.com/GICSAFePhD/RailML> (visitado 14-02-2024).
- [242] Wikipedia. "Accidente ferroviario de Versalles." (2024), dirección: https://es.wikipedia.org/wiki/Accidente_ferroviario_de_Versalles (visitado 14-02-2024).
- [243] V. Chandra y M. Verma, "Fail-Safe Interlocking System for Railways," *IEEE Design & Test of Computers*, 1991. DOI: 10.1109/54.75664.
- [244] "Fail-Safe Signalization Design for a Railway Yard: A Level Crossing Case," *IFAC World Congress*, 2010. DOI: 10.3182/20100830-3-DE-4013.00056.

- [245] Wikipedia. “Train communication network (TCN).” (2024), dirección: https://en.wikipedia.org/wiki/Train_communication_network (visitado 18-07-2024).
- [246] M. S. Durmuş, U. Yıldırım y M. T. Söylemez, “Application of Functional Safety on Railways Part I: Modelling & Design,” *Asian Control Conference (ASCC)*, 2011.
- [247] A. R. T. C. Limited. “Working of Level Crossings - Rules 1 to 11.” (2024), dirección: <https://www.artc.com.au/uploads/TA20-Section-9.pdf> (visitado 18-07-2024).
- [248] “All Aboard with Xilinx Exploring the Future of Trains and Railways,” *Xilinx*, 2020.
- [249] “SOC for Railway Train & Railways,” *Xilinx Webinar*, 2020.
- [250] AMD. “Vivado Design Suite.” (2024), dirección: <https://www.xilinx.com/products/design-tools/vivado.html> (visitado 18-07-2024).
- [251] “Arty Evaluation Board Tutorial LwIP Applications for the Arty Evaluation Board,” 2015.
- [252] INTEL. “Quartus Prime Design Software.” (2024), dirección: <https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html> (visitado 18-07-2024).
- [253] M. N. Menéndez. “Videos demostrativos.” (2024), dirección: https://www.youtube.com/watch?v=CXtid3R2SAg&list=PLZ0XnTXa7AZ2g0SDj3jUp7RSoLuD_Jkyb (visitado 18-07-2024).