Automatic generation of VHDL code from graph-based railway topologies

Martín N. Menéndez

Laboratorio de sistemas embebidos,
Universidad de Buenos Aires,
Buenos Aires, Argentina
E-mail: menendez.martin.91@gmail.com

Aportante desconocido

Laboratorio de revisiones de papers,
Universidad de revisores de papers,
Buenos Aires, Argentina
E-mail: revisor@gmail.com

Aportante desconocido

Laboratorio de revisiones de papers,
Universidad de revisores de papers,
Buenos Aires, Argentina
E-mail: revisor@gmail.com

Ariel Lutenberg

Laboratorio de sistemas embebidos,
Universidad de Buenos Aires,
Buenos Aires, Argentina
E-mail: alutenb@fi.uba.ar

Abstract: This article presents a tool implemented in Python that given behavioural description of a railway system automatically generates the code expressed in hardware description language to implement the corresponding interlocking system. For this purpose the railway topology is modeled using the geographical approach and adopting graph networks. The workflow for the generation of the hardware description language and the results obtained for topologies of different complexity are presented in detail.

Keywords: Railway interlocking system; Automatic; Graph networks; FPGA; VHDL

Reference to this paper should be made as follows: Rodríguez Bolívar, M.P. and Senés García, B. (xxxx) 'The corporate environmental disclosures on the internet: the case of IBEX 35 Spanish companies', *International Journal of Metadata, Semantics and Ontologies*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Manuel Pedro Rodríguez Bolívar received his PhD in Accounting at the University of Granada. He is a Lecturer at the Department of Accounting and Finance, University of Granada. His research interests include issues related to conceptual frameworks of accounting, diffusion of financial information on Internet, Balanced Scorecard applications and environmental accounting. He is author of a great deal of research studies published at national and international journals, conference proceedings as well as book chapters, one of which has been edited by Kluwer Academic Publishers.

Belén Senés García received her PhD in Accounting at the University of Granada. She is a Lecturer at the Department of Accounting and Finance, University of Granada. Her research interests are related to cultural, institutional and historic accounting and in environmental accounting. She has published research papers at national and international journals, conference proceedings as well as chapters of books. Both authors have published a book about environmental accounting edited by the Institute of

Accounting and Auditing, Ministry of Economic Affairs, in Spain in October 2003. dasdad
aasdasd

1 Introduction

This work was required by Trenes Argentinos (Trenes Argentinos (2019)) a state company that manages most of the Argentine railway network. The requirement was to develop a workflow to implement electronic interlocking systems in order to use them in different locations.It is an important project because tens of interlocking systems are needed by Trenes Argentinos, and the price of a single interlocking system ranges from 2 to 10 millions dollars. One of the main motivations for this project is to minimize technological dependence on external suppliers.

An interlocking railway system has the aim to avoid trains collisions and derailments, and to operate the signalling (traffic lights, barriers in the crossings paths, switches). Hence the interlocking system is a critical system, and therefore its design should be done following appropriate procedures (Banci, Fantechi and Gnesi (2005)).

In Fig. 1 the workflow of the entire project is presented. In this article the colored parts of the figure are discussed, what includes the automatic generation of the VHDL code for the interlocking system, following the geographical approach (Banci and Fantechi (2005)).

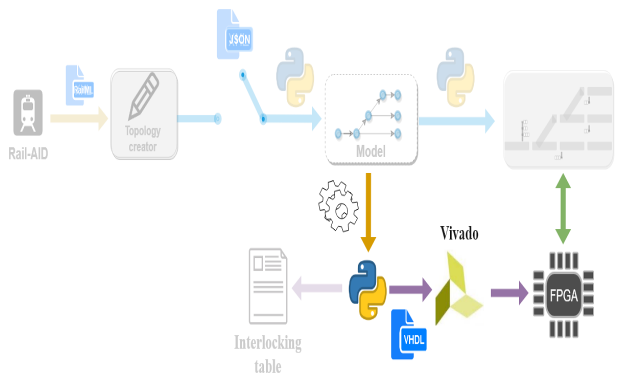

Figure 1: Workflow of the entire project. The scope of this article are the colored parts.

In the geographical approach, also called topological, the behavior of the network at a general level is determined from the behaviour of each individual element according to its relative position in the network (Sener, Kaymakci, Ustoglu and Cansever (2015); Wang, Chen and Huang (2013)). This facilitates the automation of the analysis of the network topology from which the implementation code is generated. Thus, the need for a specific analysis for each topology is avoided, since the implementation is solved from general rules (Yıldırım, Durmuş and Söylemez (2012)).

It is necessary to automate the process because there are many different locations where interlocking systems are needed. The automation is based on the railway topology and the behavioural model of the elements that compose it. The variety of elements in the network is limited and it is their arrangement in the network that gives the system its funcionality. Even small changes to the topology involve redoing much of the VHDL code. It takes considerable time to define the system manually, therefore, the workflow must be automated.

Most of companies[X] use redundant microprocessors to achieve the SIL 4 [SIL4] level required in interlocking systems. There are two interlocking implementations using PLCs: ProRail in the Netherlands and Efacec in Portugal. In this case, they are special PLCs from HIMA, which is the only company in the world that manufactures SIL 4 certified PLCs.

In recent years the use of FPGAs in the railway industry has grown. FPGA based systems have been used to integrate systems that in the past were independent and disconnected from each other [”All Aboard with Xilinx: Exploring the Future of Trains and Railways” http://twitter.com/XilinxInc/status/1283764080021569536] These FPGA-based systems offer certification capabilities with specific SIL levels. Using FPGAs there is less dependence on the manufacturer, even more in a context in which it is intended to use diversity[REF].

2 Railway Network Analyzer

To illustrate the analysis of the network and the subsequent implementation of the interlocking system we start with the topology illustrated in Fig. 2 called shunting yard. This diagram is the result obtained after performing the steps that are grayed out in Fig. 1, and that are not explained in this work.

Within all the possible topologies, the shunting yard was chosen because it has an intermediate complexity. This allows us to present different results and situations that are of interest for the analysis and will be shown in Section 4. A shunting yard is a widely used topology in stations where railway workshops are located in the vicinity and are used to withdraw or inject trains into the railway network without interrupting service.

A Railway Network Analyzer (RNA) was designed in Python to automatically analyze different topologies. The input of the RNA is a representation of the topology by means of a graph, where each track is modeled as a node and the connection between the tracks are modeled with an edge.

The RNA classifies the nodes according to their relative position in the network, as shown in Fig. 2. The nodes that have a single neighbor (blue) are classified as ’extreme’, either absolute or relative depending on whether they are located at the end of the network or are part of a neighboring network. Those with three neighbors (red) are classified as ’switch’ nodes, because they include the switches. For example, tracks B and F have switches, which allow for the bifurcation of the route.

According to the position relative to the switches, the nodes of ’direct switch’ (purple), which continue the normal circulation, and the nodes of ’branched switch’

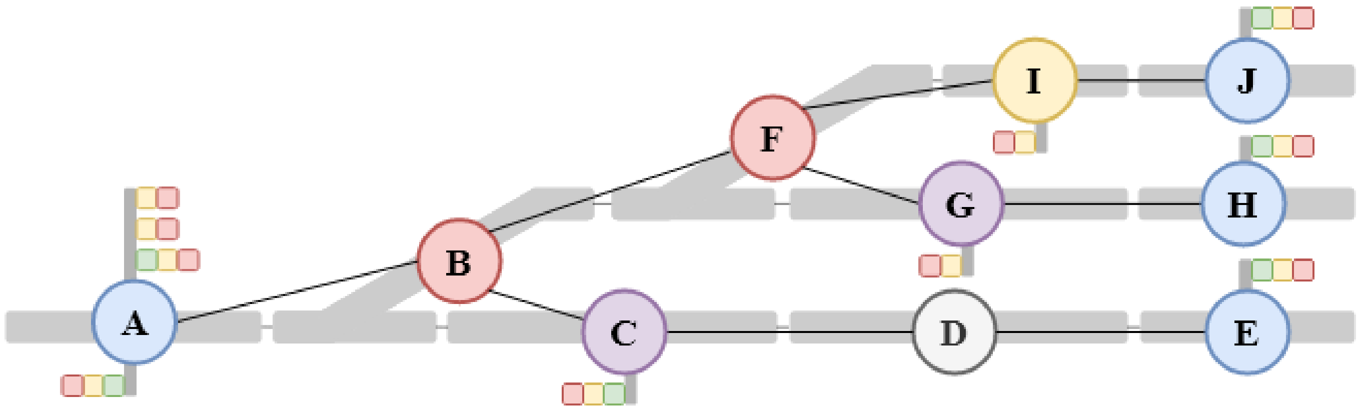

Figure 2: Shunting yard topology with graph overlapped

(yellow), which branch the route, are classified. All other nodes are 'simple' nodes (grey). The corresponding semaphores to use all the routes that the network supports, with their orientation and necessary number of aspects, are determined by the NRA (Fig. 2). All the routes generated, their conditions and effects are recorded in the interlocking table. The simplest nodes do not have semaphores, while the nodes adjacent to the switch nodes are those that concentrate a greater quantity of semaphores, according to the railway standards [12].

The RNA's algorithm will be presented in detail in a future publication to be sent to IEEE Transactions on Intelligent Transportation Systems (T-ITS) and therefore is not discussed in this article.

3 Automatic VHDL code generation

Using the output of the RNA, that is the network modelled by a graph, having all the corresponding semaphores, switches, etc. the VHDL code is generated by automatically replacing each node by a VHDL module and automatically interconnecting the modules, according to the graph provided by the RNA. In each connection to be made, it is evaluated if an immediate replacement can be made or if the modules should be adapted previously.

3.1 Railway network module

Railway network module contains the blocks that implement each of the nodes, connected as indicated in the graph, and is the core of the system as it regulates the interlocking logic itself. It is formed by nodes and switches modules whose individual behavior is complemented by their neighbors to give a safety behavior to the system.

3.1.1 Node module

A main module was designed with all the functionalities (maximum number of neighbors and signals, presence of switches and level crossings). All nodes descend from this design. The finite state machine with datapath (FSMD) of the module is shown in Fig. 3.

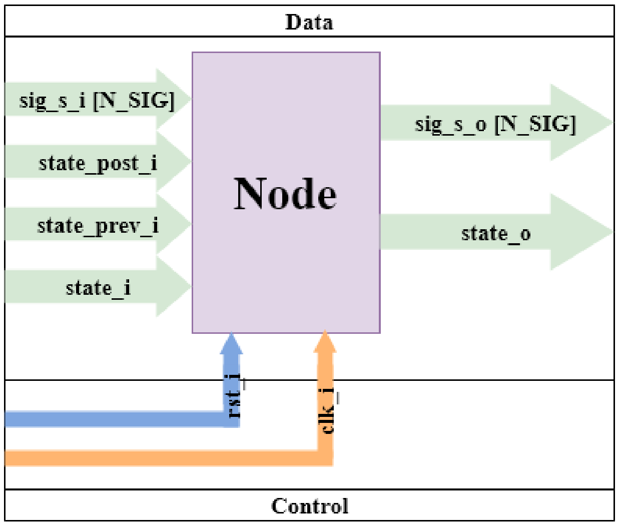

Figure 3: Mother node's FSMD

The code generator will instantiate each node, limiting the functionalities of the mother module, according to the classification given by the RNA. These instantiated nodes will be connected automatically following the topology of the graph.

3.1.2 Switch modules

The other important module in the network is the switch module that allows connections between multiple nodes and redirects signals based on the position status of the switch.The finite state machine with datapath (FSMD) of the module is shown in Fig. 4.

The node module only admits connections with two neighbors (previous and posterior), while in a topology like the shunting yard presented in Fig. 2 there are nodes that have three neighbors. To implement a generic solution with a single mother node model, a strategy will be needed to be able to represent nodes with multiple connections using simple nodes.

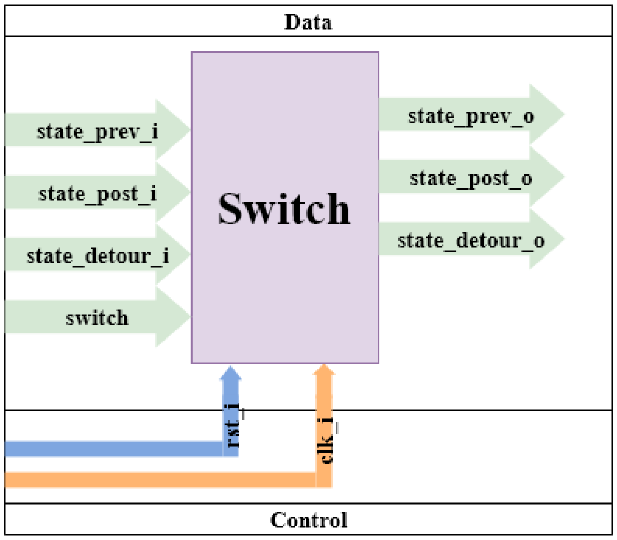

Figure 4: FSMD of the Switch module

3.1.3 Replacement algorithm

If each node classified in Fig. 2 were to be replaced by an equivalent module, only five different modules would have to be designed with very similar behaviors except for the number of ports. That is why an algorithm is proposed to perform a replacement of the change nodes, as shown in Fig. 5.

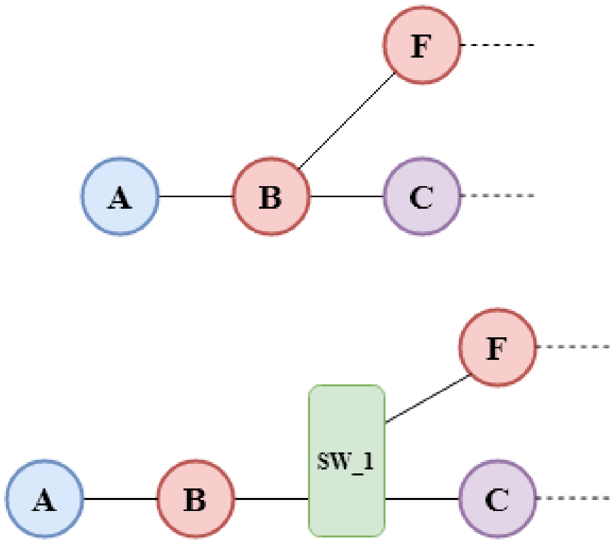

Figure 5: Replacing algorithm

In this way, the equivalent graph network of the Shunting Yard will be expanded to be able to incorporate the switch modules that will allow any node of the network to be implemented with the modules of Section 3.1.1. The extended graph network is shown in Fig. 6

This new network of expanded graphs allows to solve the implementation of the system in a description language of harwdare in a simple and order way.

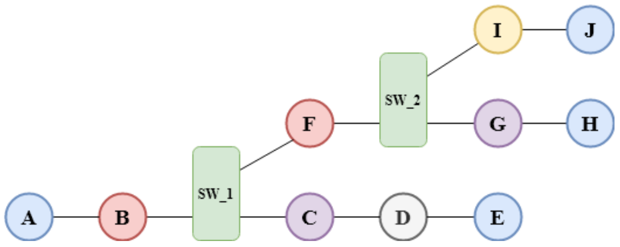

Figure 6: Extended graph network

3.2 Adapters modules

The reading of the signals of the railway system is obtained by means of sensors and communicated to the FPGA using a serial bus. Therefore, the packets must be processed and the signals distributed to the corresponding nodes and switches modules. In the same way, the outputs of the nodes and switches modules are transmitted to the semaphores and switches in the field by means of a serial communication. Thus, the corresponding signals must be gather all together and arranged in packets.

The interlocking module includes the spliter and joiner modules necessary to adapt the signals that enter the network module. That module is where the blocks representing each node and path change of the topology are instantiated together with the behavior of each traffic light and barrier.

3.2.1 Spliter module

The function of the spliter module is to receive an N-size vector of Boolean elements (packet[N]) and the order that it must be processed(process). The finite state machine with datapath(FSMD) of the module is shown in Fig. 7.

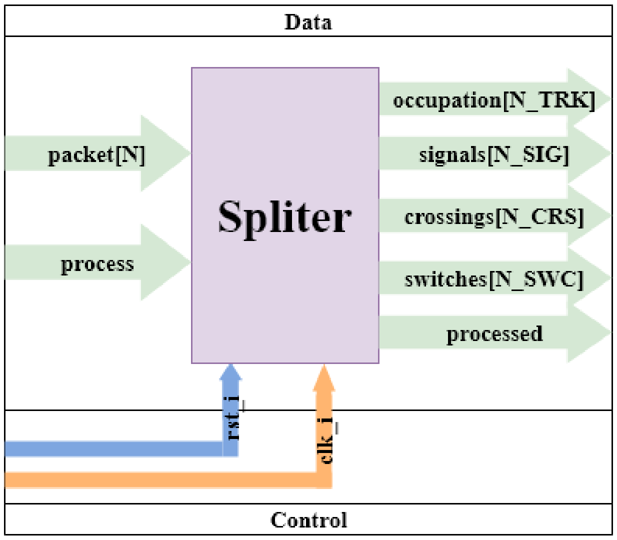

Figure 7: FSMD of the Spliter module

3.2.2 Concatenator module

The function of the Concatenator module is to regenerate an M-size vector of boolean elements (packet [M]) combining the signaling states already processed by the network module. It also returns the signal indicating that the packet has already been processed. The finite state machine with datapath (FSMD) of the module is shown in Fig. 8.

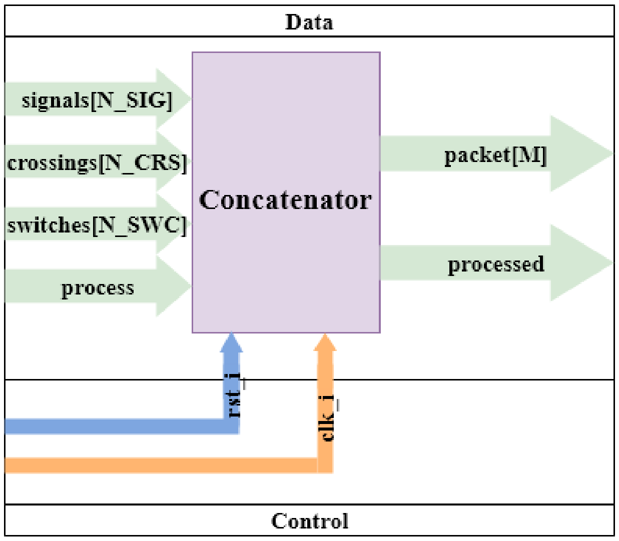

Figure 8: FSMD of the Concatenatorr module

3.3 Communication modules

Communication with the FPGA is done through a serial port connection, sending a frame of the form "<Data>". Where '<' and '>' are the beginning and end characters of the frame respectively and "Data" is a sequence of zeros and ones. This sequence begins with the state of occupation of all the tracks, followed by the signal aspect and the position of switches and barriers.

3.3.1 Detector module

The function of the detector module is to receive a sequence of characters and to assemble an output vector of boolean elements. The finite state machine with datapath (FSMD) of the module is shown in Fig. 9.

The UART sends sequentially a character (r_data (8 bytes)) and a pulse (r_available) to inform that a new data has been sent, in addition the signal N indicates the number of characters that will be sent. The detection's FSM is illustrated in Fig. 10.

The start character of the frame('<') causes a transition to the READ state in which up to N characters will be received and the next character is analyzed. If it is the end of frame('>'), then the packet is sent along with its validation (FINAL state), otherwise it goes to the ERROR state. In both cases, a new reception is expected (BEGIN state).

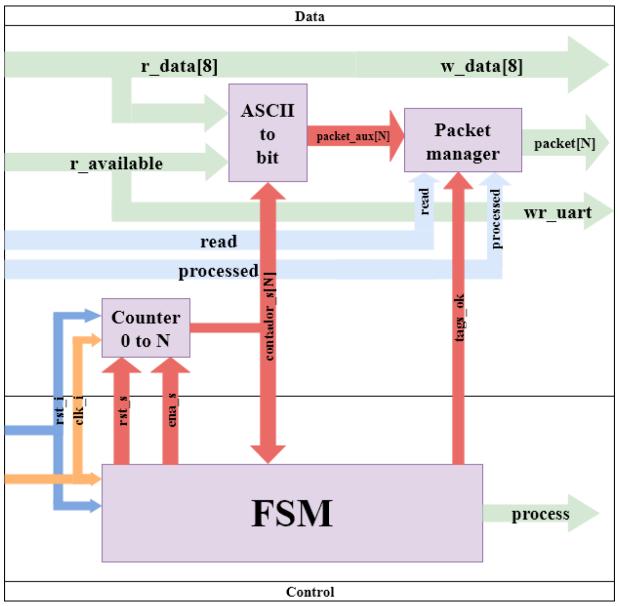

Figure 9: FSMD of the Detector module

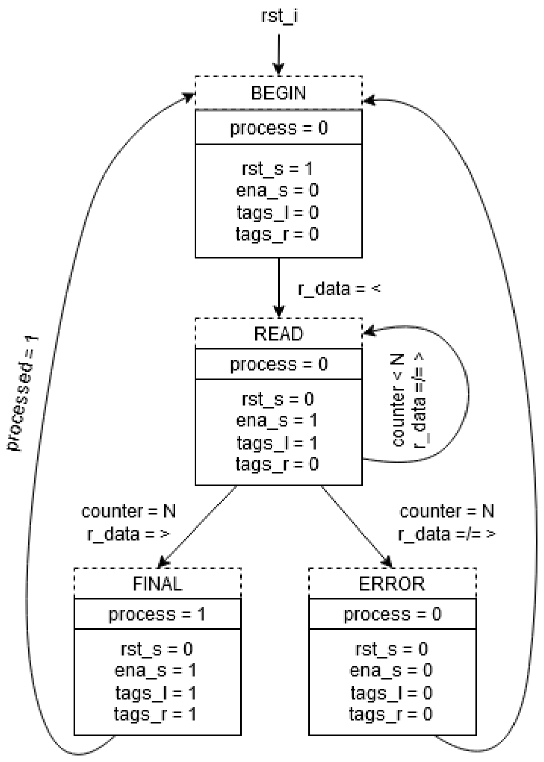

Figure 10: FSM diagram of the Detector module

3.3.2 Register module

The function of the register module is to generate M characters ('0' or '1') based on the boolean value read from packet[M], and send them to the UART. The finite state machine with datapath (FSMD) of the module is shown in Fig. 11.

The multiplexer selects each element of packet[M] according to the current value of the counter that

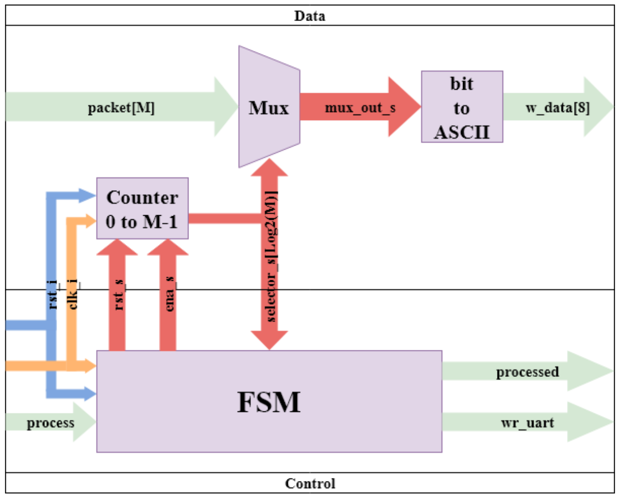

Figure 11: FSMD of the Register module

is incremented every two clock pulses. The FSM is illustrated in Fig. 12.

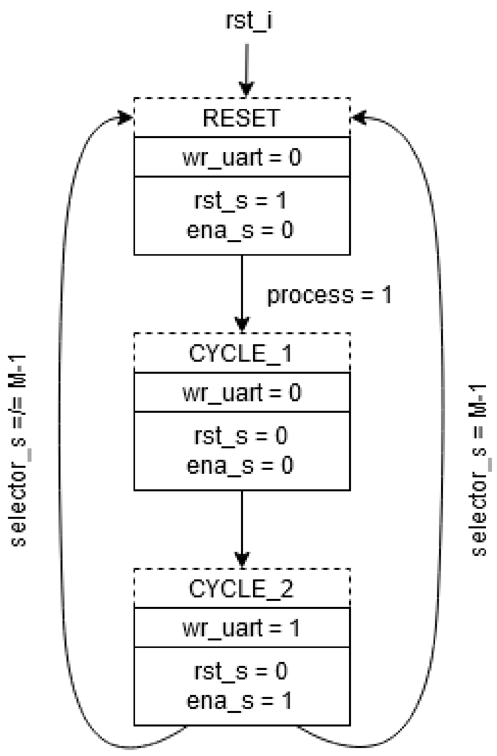

Figure 12: Register's FSM diagram

According to the logical value read in the vector ('0' or '1'), its equivalent character ("0" or "1") will be sent along with the "wr_uart" signal to indicate to the UART that this data must be stored in the output FIFO and the "processed" signal to inform the detector module that it can receive new data frames.

3.3.3 UART module

The function of the UART module is to receive the occupation and signaling data frames through the serial port, calculate the size of the frame and send it to the detector module. Then it must receive the result from the regiter module and send it back via serial port in a frame of the same format.

The input signals are of size N and the output signals are of size M. Where M is the size of all the signaling elements and N also includes the occupation signals. Therefore, it is always true that $N > M$ and it was decided to take advantage of that to instantiate the reception (UART_RX) and transmission (UART_TX) blocks automatically with sizes according to the signals they will receive. In this way FIFO_TX can use up to 25% less resources than if it had been instantiated with the same parameters as FIFO_RX.

3.4 Design proposed

All models presented are automatically connected as shown in Fig. 13. The network module is the central module where each of the nodes, switches, crossings and semaphores are instantiated along with the relationships between them. The spliter and concatenator modules serve to adapt the signals to and from the network module respectively. Finally the detector module checks that the frames are correct before sending them to the interlocking module and the register module reconstructs the frame to be transmitted.

The code generator will automatically resize the vectors of each annex module and the FIFOs of both UARTs, reinstating all the modules, including the network module, according to the complexity of the topology.

4 Case study: Shunting yard

This section will delve into the shunting yard topology presented in Section 2 after instantiating the modules in VHDL following the process discussed in Section 3.The aim is to show that the resulting implementation behave according to the rules that all interlocking systems must comply to be considered safe:

- Each track can only have one train at a time.
- To enable a route it must comply:
  - All tracks involved must be free.
  - All switches involved must be in the correct position.
  - All barriers involved must be down.
  - There can't be conflict routes enabled.
- An enabled route is indicated by its initial green/yellow signal as appropriate.

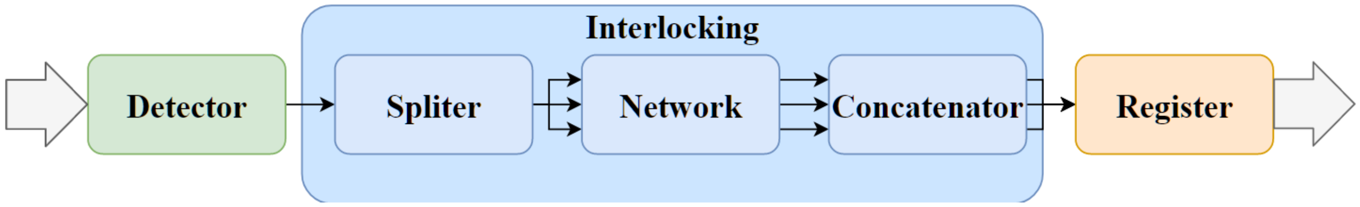

Figure 13: Blocks diagram

In an interlocking system, ”route” is defined as the path between two consecutive signals in the same direction. The RNA detects the maximum number of routes that the topology supports and specifies them in a technical document of vital importance in the railway industry: the interlocking table. The interlocking table (Table 1) was obtained automatically using the RNA algorithm.

Table 1 Interlocking table generated

Route	Begin	End	Tracks	Switches	Conflicting routes
R_1	1	10	B/C/D/E	B:Normal	R_2,R_3,R_4, R_5,R_6
R_2	2	9	B/F/G/H	B:Reverse F:Normal	R_1,R_3,R_4, R_5,R_6
R_3	3	8	B/F/I/J	B:Reverse F:Reverse	R_1,R_2,R_4, R_5,R_6
R_4	4	7	B/A	B:Normal	R_1,R_2,R_3, R_5,R_6
R_5	5	7	F/B/A	B:Reverse F:Normal	R_1,R_2,R_3, R_4,R_6
R_6	6	7	F/B/A	B:Reverse F:Reverse	R_1,R_2,R_3, R_4,R_5

This table defines all the possible routes that can be enabled in the topology and the conditions that must be met to enable them. For example, Fig. 14 illustrates route 1 (R_1) that begins at signal 1 and ends at signal 10. The route requires that the switch in B be in normal position and that the tracks B, C, D and E are free. In addition, the initial signals of the other conflicting routes must be red in order to not allow other trains to collide with the train that is using route 1 at that time.

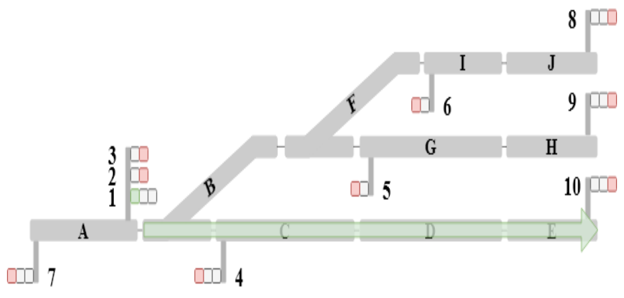

Figure 14: Route 1

All other routes defined in the interlocking table were verified to be safe and their behavior was as expected. Any other route imaginable will be a compound route based on the basic routes defined in the table. In this topology each route is in conflict with all the others, but generally a route is in conflict only with some opposing routes or that share the use of the change machine.

In Fig. 15 the automatically generated network is presented, where each of the ten nodes and the two change machines were instantiated in twelve modules of different sizes. The difference in size between the modules is due to the fact that the classification of the nodes implies a different allocation of resources from the mother module.

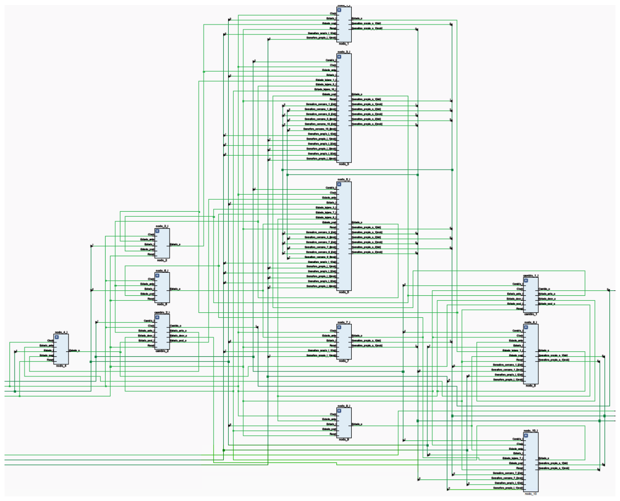

Figure 15: Automatically generated network

The chosen hardware platform was the Arty Z7-10 which has 17600 LUTs, 35200 Flip-Flips and 100 input/output ports. After synthesizing and implementing the system to generate the bitstream necessary to record in the FPGA, the amount of system resources used is obtained, as shown in Table 2.

Table 2 Resource usage in FPGA

	LUTs	FFs	IOs	BUFG
Available	17600	35200	100	32
Used	624	1340	16	1
%	3.55	3.81	16	3.13

As can be seen from the table, the use of resources does not exceed 4% of the total available on a low cost platform. This indicates that, even maintaining the current platform, it would be possible to implement more complex and secure systems.

5 Study of other topologies

Similar tests were carried out for different railway topologies, with different complexities and number of elements, as shown in Table 3. In all cases, the operation of the system implemented automatically in the FPGA was according to the corresponding interlocking table.

Table 3 Detail of generated topologies

Topology	#Tracks	#Signals	#Switches
"Y"	4	4	1
Bypass	10	10	2
Shunting yard	10	10	2
Simple station	14	16	4
Hub station	24	27	12
Terminal station	33	36	14

Both the bypass and the shunting yard topology have the same number of elements, but the former is less complex. As the number of tracks in the topology increases, a comparable number of semaphores is required based on the RNA results, but the number of switches will depend on the network.

The two most complex topologies are the hub station and the terminal station. The first is used in stations with many platforms where different railway lines must converge. While a terminal station is usually the beginning or end of the rail network, from where different branches arrive or depart simultaneously.

Each path or switch will need one bit and each semaphore will need 2 bits to be represented in the data frame. Therefore the shunting yard topology will need an output data frame of length $M = 2 * \#Signals + \#Switches = 22$ and an input data frame of length $N = M + \#Tracks = 32$.

Using the RNA algorithm combined with the automatic code generator together with the representation in graphs of each topology presented in Table 3 we can obtain a bitstream. The use of resources used by each topology and the time it tooks to obtain the bitstream is represented in Table 4.

Table 4 Detail of implemented topologies

Topology	N	M	LUTs	FFs	time
"Y"	–	–	—	—-	-m–s
Bypass	32	22	513	1091	2m10s
Shunting yard	32	22	626	1340	2m24s
Simple station	50	36	871	1906	2m28s
Hub station	90	66	1111	2490	2m30s
Terminal station	119	86	1563	3596	2m57s

Even when the size of the frames is multiplied up to 4 times its original value, the time necessary to obtain the implemented system is between 2 and 3 minutes. Which is a notable advantage over ad-hoc designs that required dozens of hours of work. The use of LUTs triples but is still below 10% of the total LUTs on the platform and the use of Flip-Fliops is four times with similar conclusions. In all cases, the use of input and output ports and BUFG was kept constant by always using the same number of communication pins and accumulating the data internally in the FIFOs of each UART module, instantiating their sizes depending on the complexity of the topologies.

6 Conclusion

Taking advantage of the wide variety of topologies analyzed by the RNA, an automatic VHDL code generator was successfully developed that manages to implement both simple and complex topologies in a single FPGA system. The design and implementation process is reduced from several weeks to a few hours. Their behavior was as described in their respective interlocking tables, also generated automatically.

In future stages of the project, it will be sought to redound the most critical processes and diversify the hardware platforms to reduce the probability of system failures and increase availability. The communication modules will be updated to allow industrial ethernet and WiFi connections to have access to a cloud platform to report system data. In parallel we are working on the verification and validation of the entire system for a prompt certification.

References

Trenes Argentinos (2019) 'Trenes Argentinos operaciones', *https://www.argentina.gob.ar/transporte/trenes-argentinos.*

Banci M., Fantechi A., Gnesi S. (2005) 'Some experiences on Formal specification of Railway Interlocking Systems using Statecharts'.

Banci M., Fantechi A. (2005) ' Geographical Versus Functional Modelling by Statecharts of Interlocking System', *Electronic Notes in Theoretical Computer Science*, 133, pp.3-19.

Yıldırım U., Durmuş M.S., Söylemez M.T. (2012) 'Automatic Interlocking Table Generation for Railway Stations using Symbolic Algebra', *IFAC Proceedings*, Volumes, 45(24), pp.171-176.

Oz, M., Sener, I., Kaymakci, O., Ustoglu, I. y Cansever, G. (2015) 'Topology Based Automatic Formal Model Generation for Point Automation Systems', *Information Technology And Control*, 44(1)

Wang, D., Chen, X. y Huang, H. (2013) 'A graph theory-based approach to route location in railway interlocking', *Computers Industrial Engineering*, 66(4), pp.791-799.