# TCN (Train Communication Network) gateway for simulation

JAIME JIMÉNEZ, J. L. MARTÍN, JAGOBA ARIAS, UNAI BIDARTE, ARMANDO ASTARLOA
Departamento de Electrónica y Telecomunicaciones
Universidad del País Vasco
Alda. Urquijo s/n, 48.013 Bilbao
ESPAÑA
jtpjivej@bi.ehu.es    http://det.bi.ehu.es/~apert/APERT_engl.htm

*Abstract:* - In this paper a behavioral model of the gateway for the Train Communication Network (TCN) is presented. It has been used in a specific verification tool for TCN devices based on a commercial VHDL simulator. By means of bit level exhaustive simulation in this tool, electronic designs in VHDL language have been validated before prototypes are produced. In this way, a virtual communication network is composed of various TCN device models which interchange pieces of information. These data and the network management parameters have been written in the configuration files of every model through the user interface. Such virtual nodes generate VHDL signals that simulate real traffic of Master and Slave Frames. However, the whole description has not been edited in plain VHDL. Upper level functions have been written in C++; these communicate with the bus controller in VHDL by means of the FLI, a special interface for this purpose. In the case of the gateway, the model consists of some blocks which are common to more simple devices, and a specific one: the TCN gateway function. In addition, an application module must have been inserted in order to produce the message traffic.

*Key-Words:* - Testbench, Train Communication Network, gateway, virtual network, simulation tool, validation process.

## 1 Introduction

In electronic design, verification is one of the most important tasks during the system project [1]. Validating a design involves guaranteeing that the product works as it was expected to [2]. Absolute validation is not viable until physical prototype is produced and thoroughly tested in real world operation. Nevertheless, this practice carries significant drawbacks: errors and bugs cannot be found until prototype is generated and, subsequently, their resolution means a new complete design and prototype process. So time to market gets longer and project costs rise in opposition to management's basic rules.

Practical validation must look for more efficient methods. Hence, design flow is divided in consecutive steps in order to produce a more and more detailed description, or model, of the system. Then a more flexible validation, but efficient indeed, is reached by verifying whether the subsequent models behave like the previous ones. This process involves specifying carefully an accurate start point which reflects exactly the behavior of the final target. This first description is usually the requirements document, where the designer tries to collect all user's needs.

Therefore, some different models describing the same definitive system are created and verified before the prototype is produced (Fig. 1). The first of them is usually the behavioural description, second the register transfer level (RTL) model and third the gate netlist (synthesis) [2]. It must be verified whether each of them matches initial specifications, and a description cannot be generated until previous one has been approved.

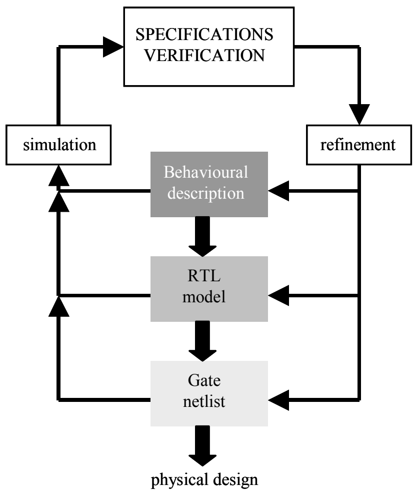


Fig. 1. Design flow for an electronic system.

Such a design flow requires specific testbenches in order to validate each model. These validation environments simulate the stimuli which the final circuit will find in real operations. Editing these testbenches is a complex task, specially when dealing with devices in communication networks [1]. In this case, although there are often few inputs and outputs, signal values can change randomly and fast, so possible stimuli may be almost infinite. Of course, the designer is not interested in all the cases but in a limited set of them. However, deciding which inputs will be forced, generating and controlling them, simulating the responses and analyzing results are as tedious as crucial tasks.

In this paper we report a specific solution to such a problem: a testbench to verify digital communication circuits for an industrial data network, used in trains [3]. The simulation environment requires at least two buses, with one bus administrator and one slave node each. It will let the designer validate a device described in VHDL language. If this model code to be tested can be synthesized by a compiler, the result is the verification of our FPGA design before prototype is produced [4].

## 2 The Train Communication Network (TCN)

An on-board train communications system had been widely demanded for modern railways, so device interoperability, distributed control architectures and integration with other external networks are made easier by interconnecting all the electronic subsystems [5-6]. The definitive version of such a standard was approved as Train Communication Network (TCN) [7]. The general architecture of TCN includes two bus types (Fig. 2):

· MVB (Multifunction Vehicle Bus), which is used for attaching the electronic equipment inside a train vehicle.

· WTB (Wired Train Bus), which is used for interconnecting the different vehicles of a train.

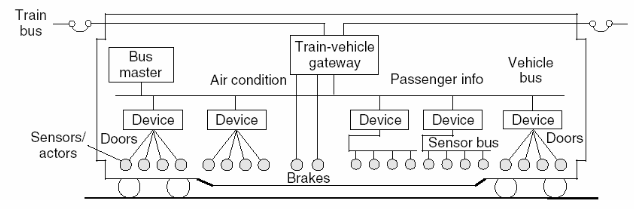


Fig. 2. Train Bus and Vehicle Bus.

Both bus types have Master-Slave architecture to control access to the network. So each vehicle bus and train bus has one Master node and several Slave ones. This type of architecture ensures time-critical data to be transferred in real time, which is an important requirement in traction control.

In a Master-Slave architecture, the Master is a device which spontaneously sends information, a Master Frame, to a number of Slave devices. It gives a Slave the permission to transmit one Slave Frame before a certain time-out. The Slave is a device that receives information from the bus or sends information through it in response to a request from the Master [7-8].

In MVB different classes of devices can be used in order to carry out various functions and services both for the vehicle and for the bus itself. Table 1 shows all the MVB device classes and their capabilities. For instance, class 1 devices play the Slave role, receive Master and Slave Frames and, when polled, send a Slave Frame. Such a device is able to transmit its device status as well as Process Data when polled and to receive Process Data from other devices [9].

TABLE 1
MVB DEVICE CLASSES AND CORRESPONDING CAPABILITIES

| | **Device_Status** | **Process_Data** | **Message_Data** | **User_Configurable** | **User_Programable** | **Bus_Administrator** | **TCN_Gateway** |
|---|---|---|---|---|---|---|---|
| Class 0 | | | | | | | |
| Class 1 | ● | ● | | | | | |
| Class 2 | ● | ● | ● | ● | | | |
| Class 3 | ● | ● | ● | ● | ● | | |
| Class 4 | ● | ● | ● | ● | ○ | ● | |
| Class 5 | ● | ● | ● | ● | ○ | ○ | ● |

●: Compulsory cap. ○: Optional cap.

### 2.1 The TCN Gateway

The fig. 3 shows the functional architecture of the TCN gateway. It must route both, the Process Data traffic and the Message Data one, from a certain MVB bus to a WTB node, which will forward it to another MVB bus. The Process Data management is simple: the gateway identifies which variables must be transferred from one bus to the other, by means of an exportation list.

For the purpose of interchanging the Message Data, a specific protocol must be followed in the network, transport and session layers. In this way, different types of messages have been established; first of all, information packets: data (DT), acknowledgement (AK) and negative acknowledgement (NK). The other ones are: connect request (CR), connect confirm (CC), disconnect request (DR) and disconnect confirm (DC) for the point to point communication; and broadcast connect (BC), broadcast data (BD), broadcast repeat (BR) and broadcast stop (BS) for the broadcast communication.

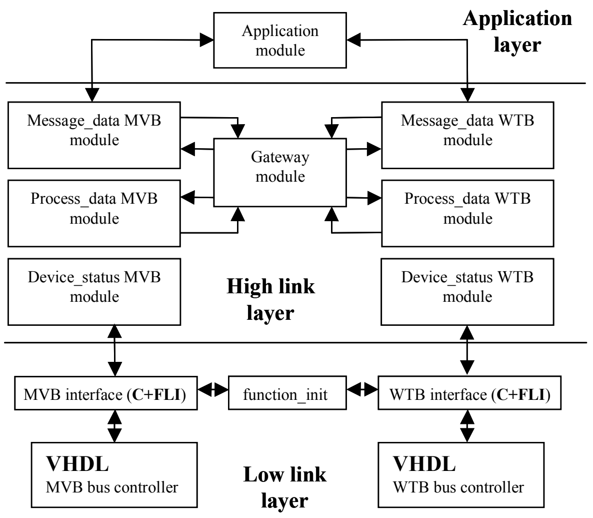


Fig. 3. Functional block diagram of the gateway.

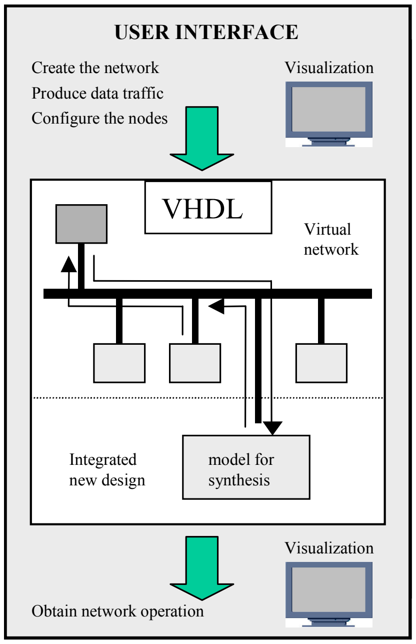


Fig. 4. Description of the simulation tool.

Fig. 3 also shows that the bus controllers have been described in VHDL and the higher level functions in C. Some special procedures, the FLI [10] routines, are in charge of interfacing both parts.

## 3 The Testbench: a Virtual Network

As fig. 4 indicates, the proposed testbench includes a user interface in order to adapt the verification environment to MVB specific requirements and a VHDL interface for integrating the design under test, as well as facilities to add this one to the virtual network [11].

In fig. 5, the languages used to describe the gateway can be associated with functional parts. The C module not only interacts with the bus controller in VHDL but also it reads the input configuration file and writes the output report one.

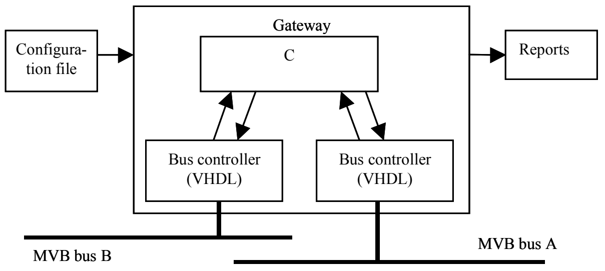


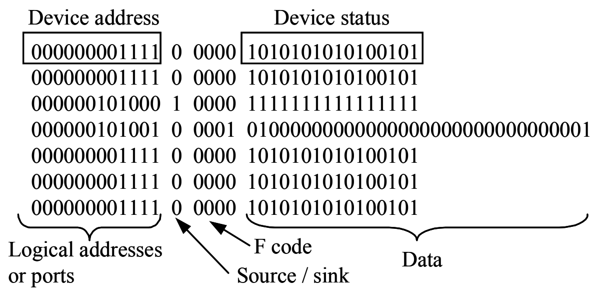
Fig. 5. Languages used to describe the gateway and interactions with input-output files.

### 3.1 The Configuration File

A piece of a configuration text file can be seen in fig. 6, showing subscribed addresses for Process Data. Here, each line contains one logical address, or port, a flag to say whether it is a source or sink, its F code, which specifies the data length, and finally all data bits. In the first line, the address belongs to the device itself and data are the Device Status. Of course, our tool is in charge of generating such a file from the user interface in a friendly way. Later, the model itself will modify the data bits upon received Slave Frames.

### 3.2 The Gateway Model

So far, this verification tool has been used to test only MVB devices. Therefore, the gateway model has been more simple than in the most general case, as fig. 7 shows. No WTB module or interface has been included, since they are not necessary at all for that purpose. However, interconnecting two MVB buses is crucial in order to validate the Message Data management.



```
Device address          Device status
000000001111 0 0000 1010101010100101
000000001111 0 0000 1010101010100101
000000101000 1 0000 1111111111111111
000000101001 0 0001 0100000000000000000000000000000001
000000001111 0 0000 1010101010100101
000000001111 0 0000 1010101010100101
000000001111 0 0000 1010101010100101
```

Fig. 6. An example of the configuration file.

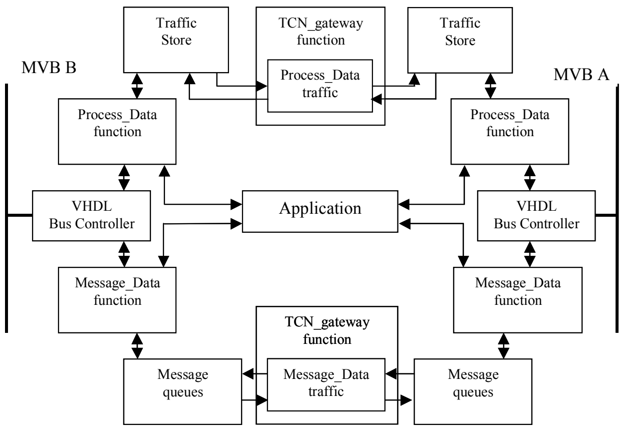


Fig. 7. The gateway used to verify MVB devices.

## 3.3 The Bus Controller

The bus controller in VHDL is represented in fig 8. It is composed of three entities and, on the right hand, in addition to the 3 clock input signals, the interface signals with the C block can be seen. These six signal are read by the FLI subroutines.

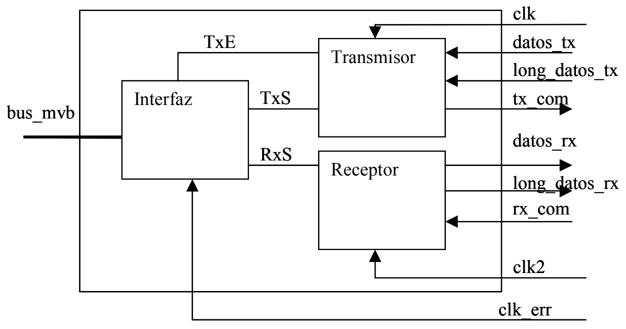


Fig. 8. The bus controller in VHDL.

## 3.4 The Test Scenario

The specific test scenario is shown in fig. 9: two MVB buses are interconnected by a gateway, the class 5 device. Each bus must have a Master Frame generator (Gen_MF), a simplified class 4 device, and, at least, one class 2 device. Sometimes, this last one may be just a class 1 device, since the Message Data capability is not needed to monitor Process Data traffic.

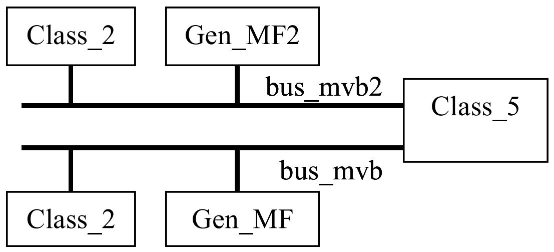


Fig. 9. The basic test scenario.

## 3.5 The Block Diagram of the Class 5 Device

The functional architecture of the class 5 device, the gateway, is represented in the block diagram of the fig. 10. If the "Gateway module" is deleted, it results in a class 2 device, and if, in addition, the Message Data module is also suppressed, it turns into a class 1 device.

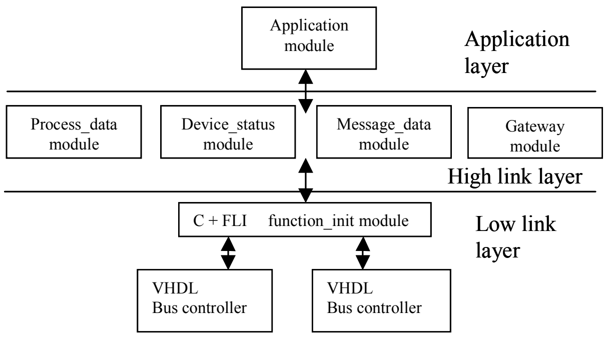


Fig. 10. The block diagram of the gateway.

## 3.6 The Master Frame Generator

In the case of the class 4 device, the Master Frame generator, the behavioral architecture is quite different, as fig. 11 shows. There are two main management blocks synchronized by a timing control module. The first one generates Master Frames for Process Data and the second one dispatches events. As a matter of fact, this last procedure is the way to send Message Data.

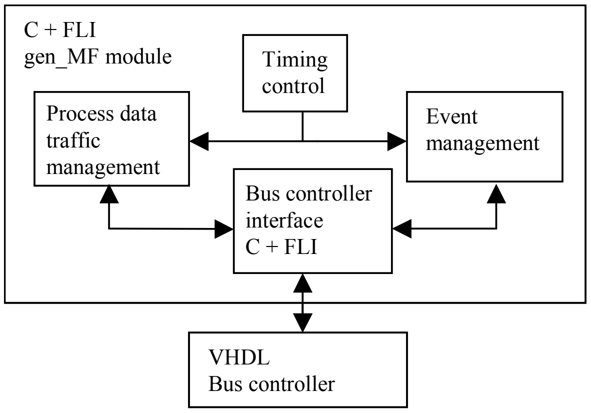


Fig. 11. The block diagram of the Master Frame generator.

## 4 Simulation Results of the Gateway

In order to validate the model of the gateway, a specific set of test cases has been simulated with some input vectors. The first test case describes the Process Data traffic between two MVB buses (fig. 12). The results have been successful and they have been represented graphically in fig. 13. The gateway has forwarded the data in sink ports of the Traffic Store in the first MVB bus to the proper source ports of the second traffic memory, following the instructions in the exportation list. The latter has been specified by means of the gateway configuration file.

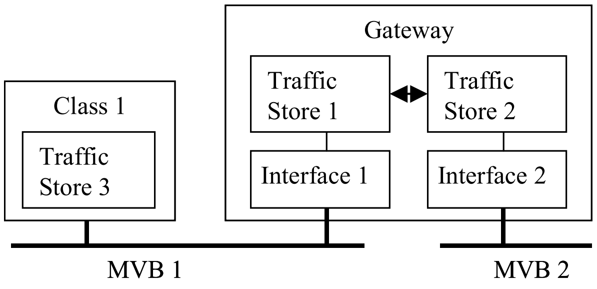


Fig. 12. The test scenario for Process Data traffic.

First, the gateway receives a Process Data Slave Frame for one of its sink ports, as any other Slave device in the MVB bus 1.

Second, it writes the data in the proper address and looks for a destination port in the memory 2, reading the exportation list. And third, a piece of received data (one of the variables in that port) is written in the Traffic Store 2.

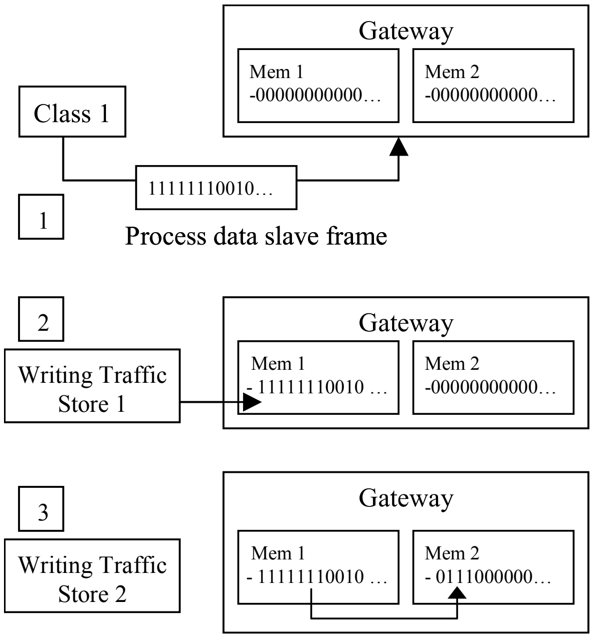


Fig. 13. Simulation results for the Process Data traffic.

The second test case is in charge of the Message Data traffic (fig. 14). Two Slave devices of class 2 will exchange messages from MVB bus 1 to the other, through the gateway. To be precise, the function 5 in the first device will send some data to the function 6 in the second one. The first time the communication ends successfully and it is represented in fig. 15. Initially, both devices establish the connection and negotiate the credit (how many packets may be sent before receiving the acknowledgement, 3 in this case). Second, the producer sends 3 consecutive packets and, afterwards, the consumer confirms all of them. Finally, last data packet is sent with a flag to indicate that there are no more data and the receiver finishes the communication with an acknowledgement.

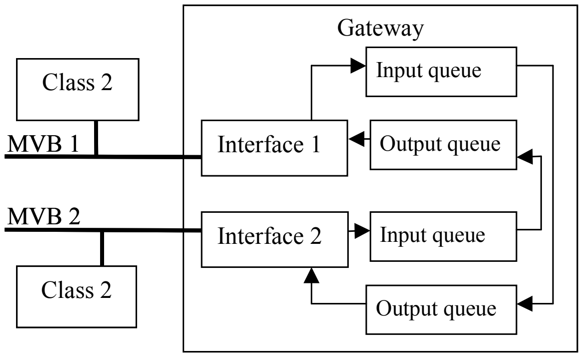


Fig. 14. Test scenario for Message Data traffic.

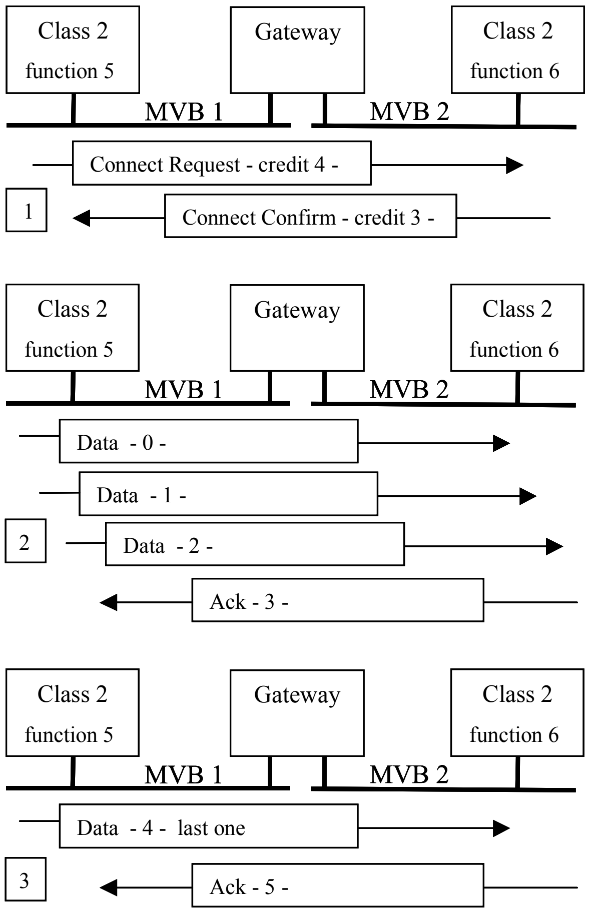


Fig. 15. Simulation results for Message Data traffic.

This has been a point-to-point connection. Another Message Data communication has been simulated to verify the broadcast emission of packets. And last, a refused connection request was tested successfully (fig. 16).

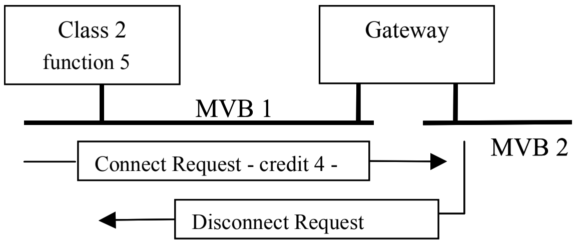


Fig. 16. Simulation of a refused connection request.

## 5 Conclusions

A behavioral model of the MVB gateway has been designed for purposes of testing by simulation. The description is composed of a VHDL module to generate the virtual electrical signals and a C one to represent the behavior. The whole model has been validated by simulating successfully 4 specific test cases, including tens of frames. Therefore a VHDL description of a class 2 device for synthesis can be verified thoroughly by means of a realistic Message Data traffic through the virtual gateway.

In this way, the specific verification tool created by the authors to validate TCN designs has been completed. In addition to the user interface, which edits the configuration files, the simulated virtual network includes interconnection to MVB buses.

## Acknowledgement

The work described in this paper has been supported by the Ministerio de Ciencia y Tecnología of Spain and FEDER in the framework of the TIC2001-0062 research project.

*References:*

[1] J. Bergeron, *Writing testbenches. Functional verification of HDL models*, Kluwer Academic Publishers, 2000.

[2] R. Rajsuman, *System–on-a-chip. Design and test*, Artech House Publishers, CA, 2000, pp. 10-11.

[3] J. Jiménez et al. Simulation enviroment to verify industrial communication circuits, *Proc. IEEE Int. Conf. Industrial Electronics, Control and Instrumentation*, IIT-07, Sevilla, 2002.

[4] J. Jiménez et al. A Top-Down Design for the Train Communication Network, *Proc. IEEE Int. Conf. Industrial Technology*, Maribor, 2003.

[5] G. Fadin, H. Kirrmann, P. Umiliacchi, ROSIN, Railway Open System Interconnection Network. Web Technologies for Railways, *Proc. Automation in Transportation*, 1998.

[6] P. Umiliacchi, The Role of European research in the railways modernisation process: The ROSIN project, *Proc. World Congress on Railway Research*, 1997.

[7] International Electrotechnical Commission, IEC 61375-1, *Train Communication Network*, 1999.

[8] H. Kirrmann, P. A. Zuber, The IEC/IEEE Train Communication Network, *IEEE Micro*, vol. 21, n. 2, pp. 81-85, 87-92, March-April 2001.

[9] A. Chavarría et al. Slave node architecture for Train Communications Networks, *Proc. IEEE Int. Conf. on Industrial Electronics, Control and Instrumentation*, Nagoya, 2000, pp. 2431-2436.

[10] ModelSim Foreign Language Interface, Version 5.5e, 2001.

[11] J. Jiménez, J. L. Martín, A. Astarloa, and A. Zuloaga, Manchester decoding algorithm for Multifunction Vehicle Bus, Proc. 2004 IEEE Int. Conf. Industrial Technology, Hammamet, 2004.