Microprocessors and Microsystems 44 (2016) 38–46

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Modern methods in railway interlocking algorithms design

Piotr Kawalec[a], Marcin Rżysko[b,*]

[a] Warsaw University of Technology, Faculty of Transport, Koszykowa 75, Warszawa, 00-662, Poland
[b] Bombardier Transportation (Rail Engineering) Polska, Ogrodowa 58, Warszawa, 00-876, Poland

ARTICLE INFO

Article history:
Received 26 October 2015
Accepted 22 November 2015
Available online 31 December 2015

Keywords:
Algorithms
Interlocking
Railways
Train control

ABSTRACT

Despite years of railway control and signalling development, modern formal description methods are still not widely used. Lack of standards in the interlocking logic construction method causes the development of the railway control systems to be more and more expensive. Moreover, the microprocessor technology used nowadays reaches its limits regarding signal processing time in decentralised systems. This forces the industry to seek for new solutions. This paper presents an algorithmic approach to interlocking logic development, together with a modern implementation methods using hardware description languages and programmable devices.

1. Introduction

Railway has become the most complex mean of transportation in terms of traffic control and safety. Currently developed solutions allow nearly autonomous train operation at very high speeds, providing continuous control on each level, from the track occupancy check, through real-time speed profile calculation and execution, to automatic routing of multiple trains in a control area. Ensuring safety of these operations requires redundant and safety-critical appliances.

Since the beginnings of the railway interlocking development the train speeds and number of required functionalities increase. Starting with the relay technology introduction, railway interlocking systems began to be decentralised and remotely controlled, so as previously one large station could have around ten manned signal boxes, since the relay era one dispatcher could fully control large area including lines and stations. It was possible also due to the fact, that the relay technology is relatively fast – the signal delay time depends mostly on electric wave propagation. Major disadvantages of the relay systems were the cost of safety-related relays and high space consumption in signal boxes.

Today most of the railway interlocking systems are based on microprocessor technology. These became more and more popular along with increasing availability of large scale of integration devices and *Commercial Off-The-Shelf* hardware. The software part of these solutions is often based on the preceding relay systems. Although it was an easy way to transfer the existing functions to a new technology platform, vulnerabilities of the predecessors are sometimes preserved. The calculation time in software solutions is also a lot higher, causing that implementing new functions becomes more problematic.

Nowadays trends also begin to point at minimising the costs of railway interlocking equipment, but keeping the safety standards at equally high level. It is therefore necessary to look for a new, complete and efficient solution for railway interlocking systems. A possible next step is to go back to the hardware, or hybrid solutions, taking all new achievements in specialised electronic devices into consideration.

Regardless of the technology used, every modern railway interlocking system has to conform to the high safety parameters. The following documents issued by the European Committee for Electrotechnical Standardization (CENELEC) describe the requirements for the new systems:

- EN50126 – Railway applications. The specification and demonstration of reliability, availability, maintainability and safety (RAMS);
- EN50128 – Railway applications. Communications, signalling and processing systems. Software for railway control and protection systems;
- EN50129 – Railway applications. Communications, signalling and processing systems. Safety related electronic systems for signalling.

These documents describe both hardware (EN50126 and EN50129) and software (EN50128) part of the system. However there are no strict guidelines for hardware logic implementation such as FPGAs. Nevertheless, due to the rapid development of these solutions, this approach needs to be revised.

* Corresponding author. Tel.: +48 698648063.
E-mail addresses: pka@wt.pw.edu.pl (P. Kawalec), marcin.rzysko@pl.transport.bombardier.com, m.rzysko@gmail.com (M. Rżysko).

http://dx.doi.org/10.1016/j.micpro.2015.11.018
0141-9331/

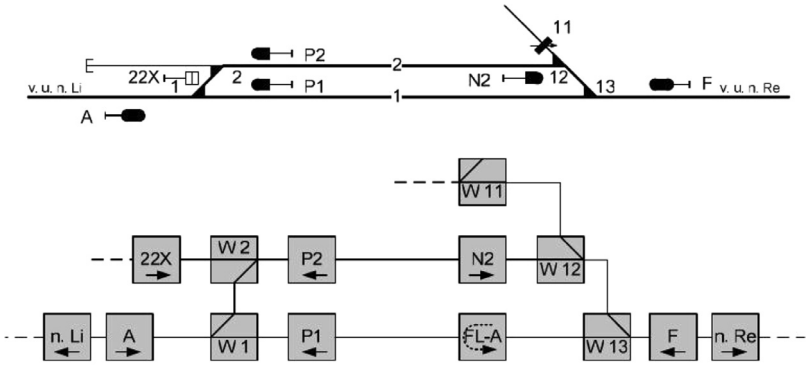

Fig. 1. Track layout and the corresponding logical objects connection plan (from [7]).

In this paper an algorithmic approach to the interlocking logic design is proposed, followed by an implementation method using programmable devices.

2. Material and methods

Many various approaches to the interlocking logic design have been developed over the decades of railway history. Some of the railway administrations established their own unique solutions. Even though there are various formal models for describing the project requirements (*Geographic Data Language* for British Railways, *PlanPro* model for Deutsche Bahn or the *SUBSET112* for European Union railways), no such model exists for defining the interlocking logic specification itself. This matter however becomes more often a topic of the scientific discussions [2,6]. Because of the recent increase of executed functions, and various tries to unify the cooperation of interlocking systems developed by different suppliers, it is even more important to create such model.

In many of the currently developed interlocking systems the *topological principle* is used. The topological principle means that every logical element type (such as point, signal, level-crossing) is described in a generic way and then used for a specific site application. In relay interlocking this can be a relay module [8], in computer-based interlocking it is usually a separate, reusable part of the code.

A basic building block of the topological system is a *logical object*. Topological principle allows us to completely describe every logical object type once, and then use such generic specification for the designed station layout. It is then necessary that the objects are organised according to the topology of the station. An example of the track layout and a corresponding logical object layout is shown in Fig. 1.

Object communicate with each other using data channels and execute the designed functions. The connections between the logical objects are usually organised into geographical and non-geographical. The geographical connections take part in all complex functions like routes, signalling, passage control. Non-geographical connections allow implementation of additional, custom functions such as blocking all points in particular area.

Taking the topological model as a base, the system core can now be described. For creating the formal specification of such model it is necessary to organise all data processed by the system into relevant vectors.

2.1. Data analysis

Considering the railway interlocking system integrally, it has two main interfaces. One between the interlocking system and the dispatching system (Man-Machine Interface), allowing the dispatcher to issue commands and observe the results. The second is between the interlocking system and the object controllers, which are connected directly to the track-side equipment (points, signals, axle counters). This interface allows us to interact with the external environment.

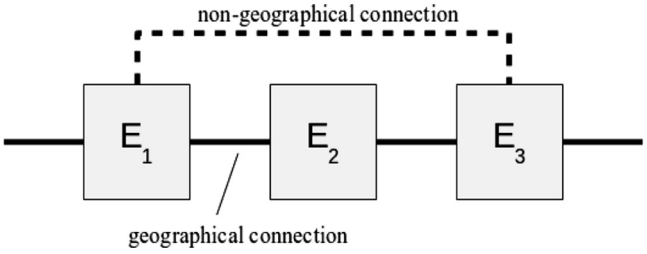

Fig. 2. Data flow between logical objects.

According to the above, data flow can be organised into following alphabets:

- input data $X = \{X_P, X_K\}$,
- output data $Y = \{Y_M, Y_S\}$.

These sets consist of the following vectors:

- command vector $X_P = \{X_{P1}, X_{P2}, \ldots, X_{PN}\}$, $X_{P\ldots} = (x_{p0}, x_{p1}, \ldots, x_{pn})$;
- check vector $X_K = \{X_{K1}, X_{K2}, \ldots, X_{KN}\}$, $X_{K\ldots} = (x_{k0}, x_{k1}, \ldots, x_{kn})$;
- indication vector $Y_M = \{Y_{M1}, Y_{M2}, \ldots, Y_{MN}\}$, $Y_{M\ldots} = (y_{m0}, y_{m1}, \ldots, y_{mn})$;
- manoeuvre vector $Y_S = \{Y_{S1}, Y_{S2}, \ldots, Y_{SN}\}$, $Y_{S\ldots} = (y_{s0}, y_{s1}, \ldots, y_{sn})$.

where N is a number of logical objects and n – number of variables in the vector.

To be able to analyse the data flow inside the system, additional vectors have to be foreseen.

When describing a single object E_i the *interlocking vector* was introduced. It allows the logical objects to communicate with each other.

- $X_Z = \{X_{ZA}, X_{ZB}, \ldots, X_{ZG}\}$, $X_{Z\ldots} = (x_{z0}, x_{z1}, \ldots, x_{zn})$;
- $Y_Z = \{Y_{ZA}, Y_{ZB}, \ldots, Y_{ZG}\}$, $Y_{Z\ldots} = (y_{z0}, y_{z1}, \ldots, y_{zn})$;

where $A, B, \ldots$ are geographical connections to neighbouring logical objects and G – non-geographical connections, as shown in Fig. 2.

Having the data identified and organised, it is possible to begin construction of the algorithm. At first it is necessary to choose the notation method.

2.2. Notation

To choose the notation method for the constructed algorithm it is necessary to ensure compact form and unambiguity.