SYNTHESIS OF PARALLEL CONTROLLERS THROUGH A LOGIC MATRIX MODEL

Camilo Quintáns*, Celso F. Silva and Enrique Mandado***

**Department of Electronic Technology, Institute of Applied Electronic*
***Department of Systems Engineering and Automatics*
University of Vigo, Spain
{quintans, csilva} @uvigo.es

Abstract: This paper gives a brief summary of the studies carried out about how to implement parallel logic controllers in reconfigurable circuits of the FPGA (Field Programmable Gate Array) kind. The necessity to base it on a mathematical model which is easily joined to the graphic editor of the designs and to the VHDL description is focused on. Also the necessity to obtain a surround of automatic systems development which integrates all the necessary elements in order to automate the whole process of development, debugging, configuration and verification is discussed. In this way a realistic system with a practical application has been achieved. The descriptive part of this study is focused on the description of the model, its validation, using an example and its test, using simulation. Finally, a system integrating all elements is described.

Keywords: FPGA, Petri nets, VHDL.

1. INTRODUCTION

Several alternatives exist to implement parallel logic controllers, for instance: based on ladder diagrams (Welch, 2000), on Petri nets (PN) (Fernandes, 1997) or based on the Sequential Function Chart (SFC) (Adamski, 1998). PN provide a unified model for both sequential (only one state can be active at a time) and parallel (several states can be active simultaneously) (Pardey and Bolton, 1991). Hence, PN are commonly used to model controllers. The theory of synchronous parallel controllers is well known and is well advanced. A good tutorials to introduce the fundamental concepts of PN are given by Murata (1984) and Zurawski and Zhou (1994). Little may be added about PN, however, advances may still be made on how to implement them in a specific technology. The designed PN are usually implemented in a programmable processor, in other words, through software.

Due to the special advantages that PN have regarding the specification of concurrent or parallel digital controllers they can be implemented in an inherently parallel technology (Wegrzy et al., 1998). In this surround digital circuits fit naturally; there is no better way of carrying out a parallel design than with a dedicated circuit. Traditionally dedicated circuits were carried out by means of printed circuit board with MSI or LSI integrated circuits or with ASIC, until the PLDs and FPGAs appeared. With these a totally parallel dedicated circuits can be made by simply configuring the logic elements (LE) in a suitable way. Since the appearance of these devices the idea of configuring them so that they implement logic controllers has arisen and several studies have been undertaken about this (Fernandes, 1997).

In order to be able to apply the reconfigurable circuit technology to the implementation of automatic controls for real applications, the automation of the design and implementation process is necessary. To achieve this, a system model and an algorithm which translates it into a hardware description, for example VHDL (IEEE std. 1076 – 2002) must be available. Once the VHDL description is obtained, the synthesis for the FPGA device to be used is carried out. When the VHDL description is done at a

structural description a more efficient synthesis can be obtained (Soto and Pereira, 2001), above all if specialised components are designed instead of directly implementing the places and transitions of the PN (Uzam et al., 2001). However, the increase in the quantity and complexity of the LE which are integrated in the FPGA devices means that nowadays it is not worth making an effort to use models based on components and due to this the use of behavioural models has become more common.

2. PERSPECTIVE

Many efforts have been made to develop translators from PN to VHDL but they do not form part of a platform which implements a complete system of development of logic automatisms (including simulation, configuration and verification) or have not lived up to expectations (Biesnack, 1993). On the other hand these translators which model PN in VHDL are usually based on the interpretation of pseudo code or a formal language based on rules (Fernandes et al., 1997; Adamski and Monteiro, 2000). These methods of modelling do not follow a mathematical notation on which one can work and which may be directly translatable to VHDL.

These translators based on intermediate languages could complicate the transformation of the model which the user has of the system to a useful model in order to implement it in VHDL, in this case means an intermediate step, whereas a mathematical model is more universal and does not add more complexity than that of the represented system. Furthermore, it is easier to generate a mathematical notation by the graphic editor than to generate a pseudo code that requires a greater abstraction level.

This study aims to give a new idea to mathematically model the PN in a way that aids the automation of the development of automatic systems of control. This idea is not exactly new (Murata, 1984), although the models used have been designed mainly to be implemented through software, in order to be able to determine their behaviour through simulation or to optimize the necessary resources (Pardey, 1992; Kozlowski, 1995).

In the following sections the logic matrix model implementing logic controllers and its integration in a platform, which implement logic controllers, are described.

3. DESCRIPTION OF THE LOGIC MATRIX MODEL

In this section the definition of the elements of the model and then the intermediate calculations to reach the general equation are shown through an example. Finally, the results obtained by the simulation and the FPGA resources consumption are presented.

3.1. Definition of the components in a Petri net description.

Terms and definitions utilized by the PN matrix model implementing a logic controller are presented in Table 1. The dimensions of the matrixes in the model are shown in Table 2.

Table 1 Definitions

CP	Current Matrix with the current marked.
NP	Next Matrix with the next marked.
T	Matrix of activation of transitions.
U	Input map.
Y	Output map.
n	Number of places.T
m	Number of transitions.
t_i	Transition i*th*, i=1,2…m.
p_i	Place i-ésimo, i=1,2…n.
f_i	Logic function of the i*th* transition, i=1,2…m.
TI	Matrix of the inputs arcs to the transitions. The coefficient $ai_{ij} \in \{0,1\}$ / $ai_{ij} = 1$ if ∃ an arc between a the place p_j and the transition t_i ; $ai_{ij} = 0$ on the contrary.
TO	Matrix of the outputs arcs of the transitions. The coefficient $ao_{ij} \in \{0,1\}$ / $ao_{ij} = 1$ if ∃ an arc between the transition t_i and the place p_j; $ao_{ij} = 0$ on the contrary.
F	Matrix of the logic equations of the transitions. The coefficient f_i is a function of the inputs variables u_i of the input map $[U]$.

Table 2 Dimension of the dot-matrix model

Matrix	Dimension
TI	$m \times n$
TO	$m \times n$
F	$1 \times m$
CP	$1 \times n$
NP	$m \times n$
T	$1 \times m$

3.2 Mapping into matrix equation model of a Petri net with n places and m transitions.

According to the above definition of a Petri net the example of the Fig. 1 is developed. In this example the number of places is *n=5* and the number of transitions is *m=4*.