The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland

IMPLEMENTING A PETRI NET SPECIFICATION IN A FPGA USING VHDL

Enrique Soto[1], Miguel Pereira[2]

[1] Dept. Tecnología Electrónica, Universidad de Vigo, Apdo. Oficial, 36200 Vigo, España, esoto@uvigo.es, http://www.dte.uvigo.es; [2] Intelsis Sistemas Inteligentes S.A. - R&D Digital Systems Department, Vía Edison 16 Polígono del Tambre 15890, Santiago de Compostela (La Coruña), mpereira@intelsis.es

Abstract*. This paper discusses how the FPGA architectures affect the implementation of Petri net specifications. Taking in consideration the observations from that study, a method is developed for obtaining VHDL descriptions amenable to synthesis, and tested against other standard methods of implementation. These results have relevance in the integration of access technologies to high speed telecommunication networks, where FPGAs are excellent implementation platforms.*

Key Words. *Petri nets, VHDL, FPGA, synthesis*

1. INTRODUCTION

In many applications it is necessary to develop control systems based on Petri nets [1]. When a complex system is going to be implemented in a small space, the best solution may be to use a FPGA.

FPGA architectures [2] are divided in many programmable and configurable modules that can be interconnected with the aim of optimizing the use of the device surface. It is necessary to remember that the main problem of PLDs, PALs and PLAs is the poor use of the device surface, that is, the low percentage of logic gates used. This occurs because this kind of programmable device has only one matrix for AND operations and other matrix for OR operations. FPGAs are different because they are composed of small configurable logic blocks (CLB) that work like sequential systems. CLBs are composed of a RAM memory and one or more macrocells. Each CLB RAM memory is programmed with the combinational system that defines the behavior of the sequential system. On each macrocell a memory element (biestable) and a configuration circuit are included. The configuration circuit defines the behavior of the macrocell.

VHDL is a standard hardware description language capable of representing the hardware with independence of its final implementation [3]. It is also widely supported by a number of simulation and synthesis tools.

2. FPGAS AND PETRI NETS

It is necessary to take into account the following points for implementing a sequential system on a FPGA:

- The system must be divided on low complexity subsystems for integrating them on each CLB of the FPGA.
- Usually, CLBs have only 4 or 5 inputs, and one or two outputs (macrocells) and sometimes it is necessary to achieve a strong division of the global system for integrating the subsystems into the CLBs.
- CLBs are interconnected between them through buses. These buses are connected to configurable connection matrices that have a limited capacity. It is necessary to bring near one another subsystems with a strong dependence between them to optimize the use of these connection matrices.

These points can be followed in many cases when implementing a Petri net. There are two kinds of elements in a Petri net: places and transitions. The circuit implementation of these elements is relatively easy, as shown in the schematics of a place and a transition in figure 1. Each one of these elements can be programmed in one or more CLBs following the model of figure 1. That would not be the most compact and efficient design, but it would be the simplest.

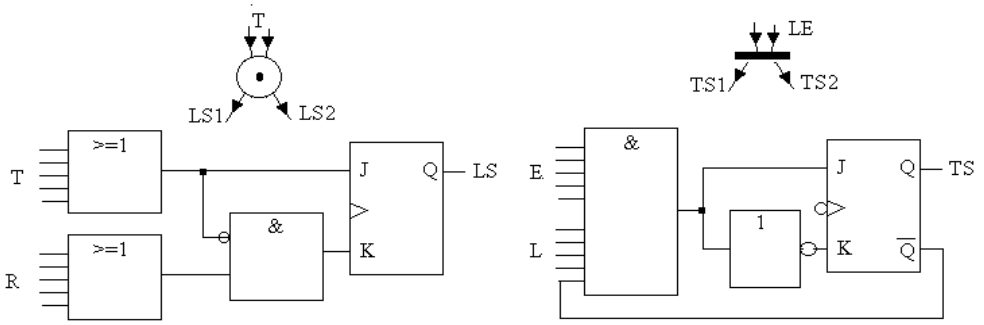

Figure 1. - Electrical schematics of a place and a transition. Each one of them can be implemented on a CLB. The main problem is the low number of inputs in a CLB. Sometimes it is necessary to use more CLBs for each element.
***T** inputs are the signals generated for the preceding transitions.*
***R** inputs are connected to the output of the next transitions.*
***LS** is the output signal of the place.*
***E** inputs are the system inputs involved in the transition.*
***L** inputs are the signals generated for the preceding places.*
***TS** is the output of the transition.*

For obtaining the most compact and efficient design, it is necessary to make the following transformations:

- In the models of figure 1, each place of the Petri net is associated to one bit-state (one-hot encoding). That is not the most compact solution because most of the designs do not need every combination of bits for defining all the states of the system. For instance, in many cases if a place is active implies that a set of places will not be active. Coding the place bits with a reduced number of bits will be a good solution because the number of

CLBs decreases. For instance, if a Petri net with six places has always only one place active, it is enough to use only 3 bits for coding the active place number (binary state encoding).

- Other transformation consists of implementing the combinational circuit of the global system, and dividing the final sequential system (combinational circuit and memory elements).

These transformations are used for making compact and fast designs, but they have some objections:

- When a compacted system is divided, may be too many CLBs have to be used, because of the low number of inputs on each CLB (4 or 5 inputs). This obstacle supposes sometimes to use more CLBs than dividing a non-compacted system.
- Verifying or updating a concrete signal of the Petri net, in a compacted system may be difficult. It is necessary to take into account the achieved transformations, and to supply the inverse transformations for monitoring the signal. This problem can be exposed in failure tolerant systems. This kind of systems needs to verify their signals, while they are running. This system may be more complex if it has been compacted previously.

To avoid the mentioned problems, this paper proposes a solution that consists in implementing the system using special blocks composed of one place and a transition. With this kind of blocks compact systems can be achieved, preserving the Petri net structure. Figure 2 shows an example of Petri net divided in 5 blocks. Each block is implemented in a CLB.

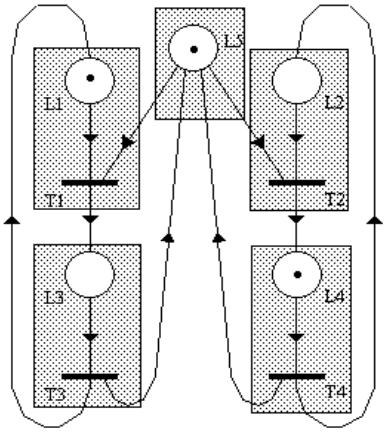

Figure 2. - Example of Petri net divided in blocks for its implementation in a FPGA.

3. IMPLEMENTATION

With this kind of implementation of Petri net based systems, every CLB is composed of a place connected to a transition. The place can be activated through its inputs connected to other transitions. It will be deactivated through reset inputs, or through the transition that is in the block (figure 3).

The transition will be active when the preceding place is active and the transition inputs have the appropriated values. Every block has two outputs, one state bit corresponding to the place, and a transition bit.

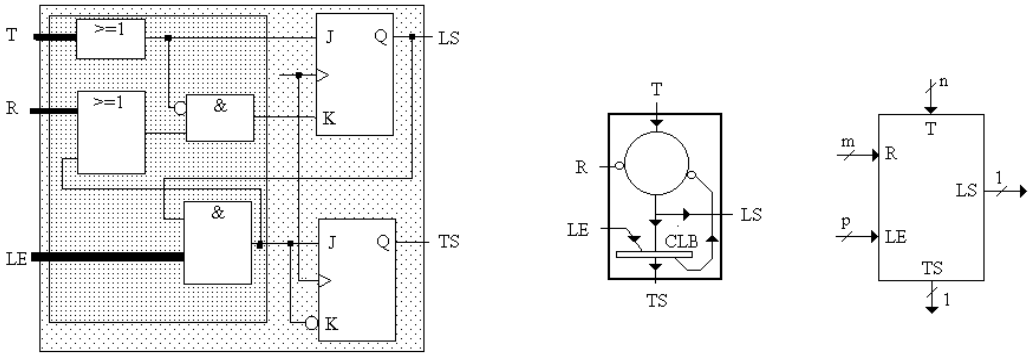

Figure 3. - Description of the new blocks that are developed in a configurable block in a FPGA. On the left, a simplified digital schematic of the block is shown.
*- **T** is the input bits set connected to other transitions for activating the place.*
*- **R** is the vector of signals for deactivating the place since other transitions.*
*- **LE** are the signals coming from other places and other input signals, that let the activation of the transition.*
*- **LS** is the output place.*
*- **TS** is the output transition.*

Figure 3 shows logical and electronic schematics of these blocks. The place and the transition are interconnected through two signals in the block. These signals are not always connected to the exterior. This detail allows a reduction of the CLBs connections in a FPGA. In many cases a concrete CLB has not enough inputs for including a block of this kind. In those cases, it is necessary to use auxiliary CLBs for implementing the block. However, it is unusual to find a Petri net on which every place is preceded by a high number of transitions (or to find a transition preceded by many places). Usually, most places and transitions in a practical Petri net are connected to one or two transitions or places, respectively (except common resources or synchronism points). Figure 4 shows some examples on which there is an element preceded by many others.

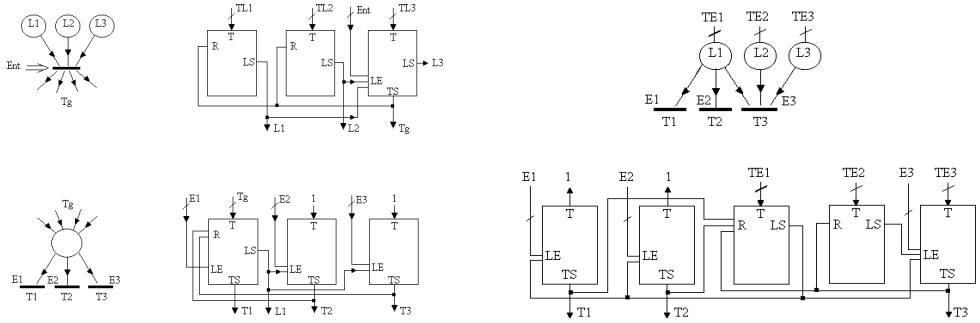

Figure 4. - Examples of different block interconnections for implementing places and transitions with multiple connections to other elements.

There are cases on which the number of CLB inputs is not enough to include a place or a transition in the CLB. Figure 5 shows a logical schematic for expanding the block inputs. In this figure, four CLBs are necessary for implementing the block. Three of them are auxiliary blocks and have the function of concentrating a number of inputs in one signal.

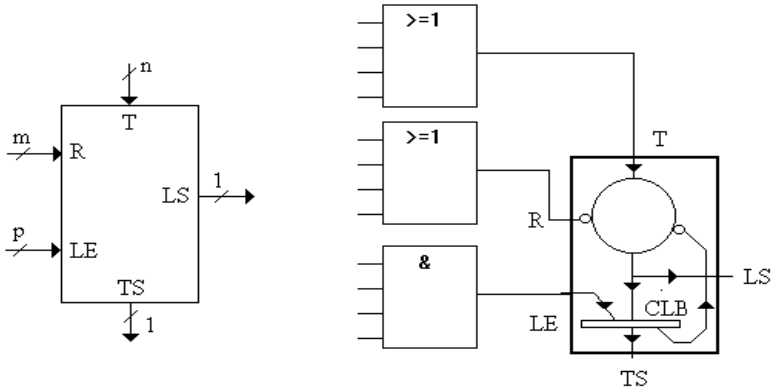

Figure 5. - Block schematic with a high number of inputs. The logic gates connected outside the block place-transition are used for incrementing the number of inputs.

4. VHDL HIGH LEVEL DESIGN

The methodology proposed uses those blocks described above as a set of parametrizable objects available in VHDL libraries. The implementation is simply the interconnection, according to the Petri net specification, of those objects whose correctness is guaranteed. The VHDL description represents correctly the specification as long the Petri net does it. The resulting architecture that is implemented within the FPGA is OHE (one-hot encoding).
This solution gives best results in the implementation of SRAM based FPGAs [4], at least as long as the number of places and the random logic associated with the transitions is not too complex relative to the combinational logic available in the FPGA.

5. EXAMPLE

Figure 6 shows the blocks interconnection of a Petri net based system on a FPGA. The example exposed corresponds to the net of figure 2.

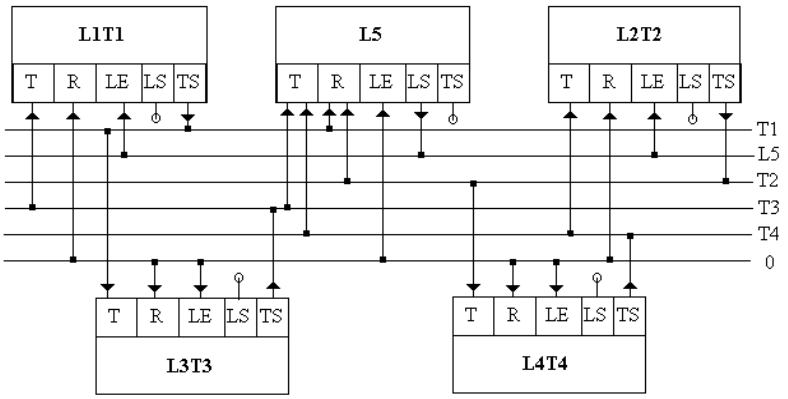

Figure 6. - Schematic of the connections for the Petri net of figure 1 with configurable blocks.

Each block is a CLB of the FPGA, and it is not necessary to include auxiliary blocks for incrementing the number of inputs of the elements of the net. There is only a place with two input signals in the net of figure 2. The rest of places have only one input signal. If some element had more than two inputs, it would be necessary to use the structures of figure 4, and then the number of CLBs would be increased. The results of different design methodologies using a sample FPGA are summarized in table 1.

Table 1

Design method	Design process	FPGA resources in use	Device Frequency Achieved
Schematic	Difficult	17 %	27,62Mhz
VDHL behavioral	Simple	21 %	16,75Mhz
This paper	Simple	12 %	63,69Mhz

6. CONCLUSIONS

In this paper, the implementation of Petri net based systems on FPGAs has been discussed. The main problem consists of using places and transitions with a different number of inputs, including the case when there are more inputs than a configurable block of a FPGA. For that, a method has been developed through two circuit models, one of them for places, and the other one for transitions. With these models a new block has been presented that contains a place interconnected with a transition. The object of this block consists in reducing the interconnections between CLBs in a FPGA, and therefore, reducing the number of inputs on each block (specially, feedback signals necessary to reset preceding places). This method is optimal for Petri nets on which most places and transitions are preceded for one or two (but no more) transitions or places. Furthermore, some possibilities have been exposed for the interconnection of blocks that increase the number of inputs in elements of a Petri net. The main purpose of this method is to integrate the maximum number of elements of a Petri net in a FPGA.

REFERENCES

[1] Zurawski, R., M.C. Zhou. "Petri Nets and Industrial Applications: a Tutorial". IEEE Trans. on Industrial Electronics, Dec. 1994.
[2] FLEX8000 HandBook ALTERA Corp. 1994.
[3] Soto, E., Mandado, E., Farina, J.. 'Lenguajes de descripcion hardware (HDL): el lenguaje VHDL'. Mundo Electronico, abril 1993
[4] Nemec, John. 'Stoke the fires of FPGA design. Keep an FPGA's architecture in mind and produce designs that will yield optimum performance and efficiency'. Electronic Design, October 25, 1994.

ACKNOWLEDGEMENTS

This work was financed by the European Commission and the Comisión Interministerial de Ciencia y Tecnología (Spain) through research grant TIC 1FD97-2248-C02-02 in collaboration with the company Versaware S.L. (Vigo, Spain).