Real-Time Control Systems Synthesis

Viorica SUDACEVSCHI[1], Victor ABABII[1], Ihor MAYKIV[2], Anatoliy SACHENKO[2], Volodymyr KOCHAN[2], Oleksiy ROSHCHUPKIN[2]
[1]*Technical University of Moldova,* [2]*Research Institute for Intelligent Computer Systems, Ternopil National Economic University, Glushkov Institute of Cybernetics, National Academy of Science, Ukraine*
svm700@mail.ru, ababii@mail.utm.md, imaykiv@yahoo.com, as@tneu.edu.ua, vk@tneu.edu.ua, o.roshchupkin@chnu.edu.ua

***Abstract* — This paper presents a synthesis method of real-time control systems based on direct mapping of Petri net model into FPGA circuit. Synchronous timed Petri nets have been developed to specify and model the control system. Switching to hardware description of the system is achieved through Hard Timed Petri nets (HTPN). Direct correspondence between the elements of the original specification and circuit components ensures that the behavioral properties and time constraints, under which activate the control system, will be respected.**

***Index Terms* — Timed Petri nets, Timed Hard Petri nets, real-time control system, FPGA.**

I. INTRODUCTION

The advantages of reconfigurable devices, including FPGAs, make them very popular during the last years. FPGAs and reconfigurable computing accelerate a lot of applications [2, 3, 6, 10]. One of the most perspective directions is the use of FPGA circuits for control systems design. The interest in this approach is based on the ability to run in parallel several control strategies and the possibility of real-time reconfiguration of the control system at the hardware-level.

One of the problems in the design of real-time control systems is the proper synchronization of all processes. Certain operations are running at predetermined time limits and processing involves time constraints. To ensure an efficient messages and data exchange is necessary to generate status and synchronization signals in strict accordance with time. In these cases, time is the basic value and time constraints require a very high accuracy that can be achieved only by using parallel or concurrent data processing techniques [1].

Implementation of parallel data processing algorithms [5] requires the verification for correctness of operation and eventual conflicts, which may lead to serious errors. For this purpose modern methods and techniques based on timed Petri nets models [4, 7] can be used.

Classical methods of real-time control systems design based on logic synthesis presents a number of disadvantages, namely high computational complexity, system specification only at low abstraction levels, resulting circuit structure does not match to the behavioral model structure. Direct mapping techniques excludes these disadvantages, which is particularly important for real-time systems in which the operations are executed according to time constraints.

In the paper is proposed a synthesis method of real-time control systems based on direct mapping of Petri net model into the hardware architecture. For this an extension of Petri nets - Timed Synchronous Petri nets (TSPN) was developed. A model for direct implementation into FPGA circuit was proposed, namely Hard Timed Petri nets (HTPN). Direct correspondence between the elements of the original specification and circuit components ensure that time constraints will be respected.

II. TIMED SYNCHRONOUS PETRI NETS (TSPN)

A Timed Synchronous Petri net (TSPN) is a 7-tuple $\left(\boldsymbol{P}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{M}_0, \boldsymbol{M}_{\max}, \boldsymbol{C}, \boldsymbol{\theta}\right)$, where:

$\boldsymbol{P} = \{ p_1, p_2, \cdots, p_N \}$ is a finite and non-empty set of places;

$\boldsymbol{T} = \{ t_1, t_2, \cdots, t_L \}$ is a finite and non-empty set of transitions ;

$\boldsymbol{A} \subseteq (\boldsymbol{P} \times \boldsymbol{T}) \cup (\boldsymbol{T} \times \boldsymbol{P})$ is a set of arcs.

The set of arcs consists of three subsets: $\boldsymbol{A} = \boldsymbol{A}^N \cup \boldsymbol{A}^I \cup \boldsymbol{A}^T$, $\boldsymbol{A}^N \cap \boldsymbol{A}^I \cap \boldsymbol{A}^T = \{\varnothing\}$, $\boldsymbol{A}^N$ - normal arcs, $\boldsymbol{A}^I$ - inhibition arcs, $\boldsymbol{A}^T$ - test arcs;

$\boldsymbol{M}_0 = \{ \boldsymbol{M}_0^{\boldsymbol{P}_1}, \boldsymbol{M}_0^{\boldsymbol{P}_2}, \ldots, \boldsymbol{M}_0^{\boldsymbol{P}_N} \}$ is the initial marking, defined as an initial number of tokens in each place;

$\boldsymbol{M}_{max} = \{ \boldsymbol{M}_{max}^{\boldsymbol{P}_1}, \boldsymbol{M}_{max}^{\boldsymbol{P}_2}, \ldots, \boldsymbol{M}_{max}^{\boldsymbol{P}_N} \}$ is the maximal marking, defined as a maximal number of tokens in each place;

$\boldsymbol{C}$ is the synchronization variable that enable the transitions firing [8];

$\boldsymbol{\theta} = \left\{ \boldsymbol{\tau}_j, \forall \boldsymbol{j} = \overline{1, \boldsymbol{L}} \right\}$ is the set of time values that specify the delays of transitions firing.

Subset $\boldsymbol{A}^N$ defines the normal arcs, through which tokens are removed from each input place and are added to each output place. Subsets $\boldsymbol{A}^I$ and $\boldsymbol{A}^T$ are necessary in behavioral analyze but do not remove or add tokens.

The weight of each arc is equal to one. Sets $\boldsymbol{P}$ and $\boldsymbol{T}$ are disjoint $\boldsymbol{P} \cap \boldsymbol{T} = \{\varnothing\}$ and satisfy the condition: $\boldsymbol{P} \cup \boldsymbol{T} \neq \{\varnothing\}$.

A transition t_j is enabled by the current marking M_k, noted $M_k[t_j>$, if and only if the relation is true:

$$V(t_j,M_k)=V_{A^N}(t_j,M_k)\cdot V_{A^I}(t_j,M_k)\cdot V_{A^T}(t_j,M_k)\cdot C\cdot\Delta,$$

where:

$V_{A^N}(t_j,M_k)=\prod_{\forall p_i\in {}^*t_j}(M_k^{p_i}\geq 1)$ - is the enabling condition in case of presence of at least one token in all input places, connected with t_j through normal arcs ($M_k^{P_i}$ - current marking in place p_i, ${}^*t_j$ - input places for transition t_j). For $A^N=\varnothing$ is considered $V_{A^N}(t_j,M_k)=1$;

$V_{A^I}(t_j,M_k)=\prod_{\forall p_i\in {}^*t_j}(M_k^{p_i}=0)$ - is the enabling condition in case of absence of tokens in all input places, connected to t_j through inhibition arcs. For $A^I=\varnothing$ is considered $V_{A^I}(t_j,M_k)=1$;

$V_{A^T}(t_j,M_k)=\prod_{\forall p_i\in {}^*t_j}(M_k^{p_i}\geq 1)$ - is the enabling condition in case of presence of at least one token in all input places, connected to t_j through test arcs. For $A^T=\varnothing$ is considered $V_{A^T}(t_j,M_k)=1$;

The firing of enabled transitions $T(M_k)$ will produce a new marking M_{k+1}, according to the following rule:

$$\forall p_i\in {}^*T(M_k)[M_{k+1}=M_k-1], i=\overline{1,N}$$
$$\forall p_i\in T^*(M_k)[M_{k+1}=M_k+1], i=\overline{1,N} \qquad (1)$$

Where: ${}^*T(M_k)$ is the set of all input places for transitions from $T(M_k)$; $T^*(M_k)$ is the set of all output places for transitions from $T(M_k)$; M_{k+1} and M_k are the number of tokens in place p after and before the transitions firing from $T(M_k)$ [9].

Petri nets allow the expression of parallel or concurrent activities in terms of transitions. Two enabled transitions in a Petri net model are concurrent if they are in causal independent relationship (they are not in conflict with each other) and therefore can fire in parallel. The level of concurrent activity in a Petri net depends on the number of enabled transition for any reachable marking M_k. Since in TSPN is possible a simultaneous firing of all enabled transitions, the level of concurrent activity will be determined by the number of fired transitions.

III. HARD TIMED PETRI NETS (HTPN)

A Hard Timed Petri net (HTPN) is a 13-tuple:

$$RPH=<T,P,A^+,A^-,A^S,A^T,A^I,P^{In},P^{Out},M_0,M_{max},C,D>$$

, where:

$T=\{t_1,t_2,...,t_L\}$, $T\neq\varnothing$ - is a set of processing elements that correspond to transition nodes;

$P=\{p_1,p_2,...,p_N\}$, $P\neq\varnothing$ - is a set of processing elements that correspond to place nodes;

A^+ - is a set of increment connections of the number of tokens in position processing element p_i;

A^-- is a set of decrement connections of the number of tokens in position processing element p_i;

A^S - is a set of state connections that determine the enable firing condition of the transition t_j related to the marking in the input place p_i;

A^T - is a set of test connections, which has the same function as the set of state connections, but the transition firing does not change the marking in the input place;

A^I - is a set of inhibitor connections, which provides an enabling function, when the place p_i stores no tokens. Inhibitor connection has the ability to test whether a place is empty. The firing does not change the marking in the input place;

$P^{In}=\{P_i^{In},\ i=\overline{1,N^{In}}\}$, $P^{In}\in P$ - is a set of places that specify the input signals to the digital system;

$P^{Out}=\{P_i^{Out},\ i=\overline{1,N^{Out}}\}$, $P^{Out}\in P$ - is a set of places that specify the output signals from the digital system;

$M_0=\{M_0^{P_1},M_0^{P_2},\ldots,M_0^{P_N}\}$ - is the initial marking, defined as the initial number of tokens in each place;

$M_{max}=\{M_{max}^{P_1},M_{max}^{P_2},\ldots,M_{max}^{P_N}\}$ - is the maximal marking, defined by the maximal number of tokens in each place;

C - is the synchronization variable.

D - is a set of processing elements, called „Delay", which specify delays between transitions enabling and firing according to time values θ.

The modeling and design of control systems are based on HTPN models. To make possible the direct mapping of these models into hardware circuit were developed the following processing elements: Position (P), Transition (T) and Delay (D) [9].

IV. MODELING AND IMPLEMENTATION OF A CONTROL SYSTEM

The sequence of operations for modeling and implementation of a control system based on HTPN models is presented in Figure 1. The diagram includes three basic components: VPNP Tool - software tool for modeling and validation of Timed Petri net models, HDL Compiler - software tool for converting a Petri net model from XML to HDL format, Quartus II Design Tool – software tool for

HDL compilation, functional simulation of the control system and FPGA configuration [11].

Control system synthesis.

Step 1. Petri net model is drawn as a set of graphical symbols of objects P, T, A and the connectivity between them using a object design entry environment of VPNP Tool (Petri Net Model Entry). The introduced Petri net model is stored in *XML* format with extension **. pn2* (Petri Net Model Library). In order to verify the behavioral correctness of the Petri net model, modeling and functional verification methods are applied. The resulted information is presented in *XML* format in *File.pn2* (XML Petri Net Model) that is used in next step.

Step 2. HDL Compiler analyzes *XML* file that contains Petri net model and extracts information necessary for HDL compilation. The extracted parameters (Parameterized Petri Net Model) is saved in *File.txt.* For HDL synthesis of a non-timed subnet functional objects *P.INC* and *T.INC* are used, for HDL synthesis of a timed subnet - *TD.INC* and *PD.INC*, respectively. The resulted HDL Petri net model is contained in *File.TDF* (HDL Petri Net Model).

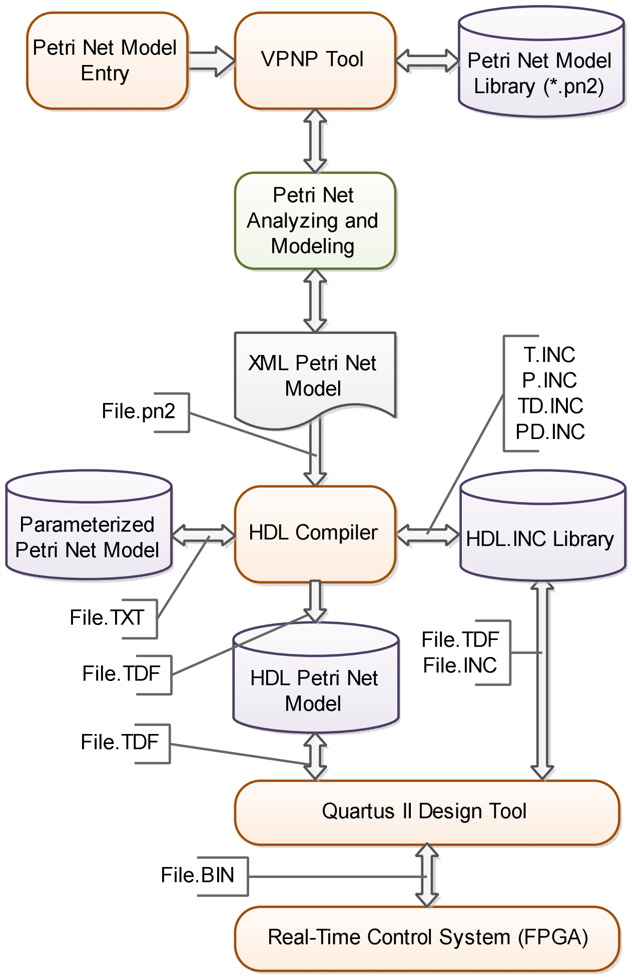

Fig. 1. Modeling and implementation of a control system.

Step 3. HDL Petri net model contained in *File.TDF* is imported into Quartus II Design Tool. After compilation time diagrams and statistics information are generated. The resulted *File.BIN* is used to configure the FPGA circuit.

V. CONTROL SYSTEM STRUCTURE

The block diagram of a control system implemented using HTPN is presented in Figure 2. The block diagram includes: *RTC* (Real Time Clock) 1.0MHz – time controller for generation of synchronization signals. In order to optimize the control system complexity, *RTC* generator frequency is selected as the lowest divider for all delays in the control system; *DPC* (Data Processing Clock) 100MHz - clock generator for data processing synchronization with 100.00 MHz frequency (can be replaced by maximum operating frequency of FPGA circuit); *FPGA HTPN* – FPGA circuit for Hard Timed Petri net implementation; $\boldsymbol{X}(t)$ Pin – control system input logic signals that determine the state of the controlled process; $\boldsymbol{Y}(t)$ Pout - control signals; HPN TC - Hard Petri net to generate time values $\boldsymbol{\theta}$ associated with transitions; *HPN PC* – Hard Petri net for $\boldsymbol{X}(t)$ processing and $\boldsymbol{Y}(t)$ generation; $\boldsymbol{\theta}=\left\{\tau_j, \forall j=\overline{1,L}\right\}$ - the set of time values specifying transitions firing delays.

The mathematical model of a control system is: $\boldsymbol{C}=\left[\boldsymbol{X}(t), \boldsymbol{Y}(t), \boldsymbol{HTPN}\right]$, where:

$\boldsymbol{X}(t)=\left\{x_i(t), \forall i=\overline{1,N}\right\}$ - is a set of input logic signals; $\boldsymbol{Y}(t)=\left\{y_i(t), \forall i=\overline{1,M}\right\}$ - is a set of control logic signals;

$\boldsymbol{HTPN}$ - Hard Timed Petri net that is implemented into FPGA circuit.

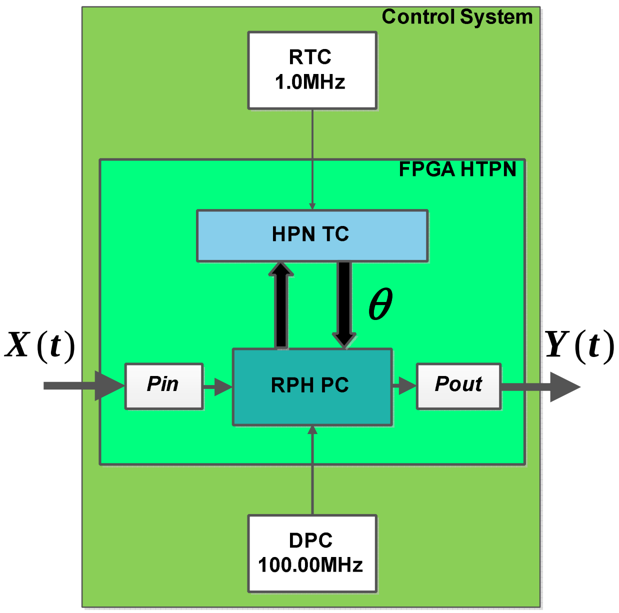

Fig. 2. Control system block diagram.

VI. REAL-TIME MODEL OF A PROCESS

Let's consider a real-time process (RTP) in which for execution of each operation a timeframe is assigned. RTP consists of a set of concurrent or parallel operations OP,

$$RTP = \bigcup_{i=1}^{N} \left(OP(\tau_i) \right) \quad (2)$$

The structure of the real-time process is presented in Figure 3, where Initial data - input data to the digital system or raw materials required for the production process, Final result - the result of the data processing or the final product obtained after technological/production process completion, $X(t)$ - the logic state of operations $OP_i, \forall i = \overline{1,N}$ and $Y(t)$ - control signals for operations fulfillment.

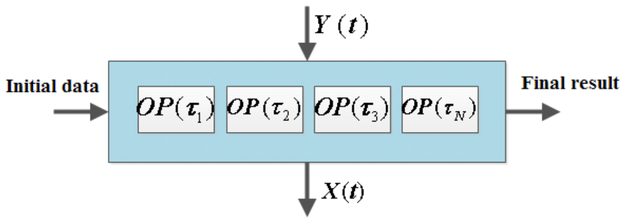

Fig. 3. Real-time process structure.

This structure can be applied to the design of concurrent data processing systems and the design of technological/ production processes.

The logical behavior of a real-time process can be described by the following model (3):

$$RTP = \begin{cases} OP_1 : IF(x_1), init(y_1), \\ Delay(\tau_1), set/reset(y_1); \\ \ldots \\ OP_i : IF(x_i), Delay(\tau_i), set/reset(y_2) \\ Delay(\tau_{i+1}), reset/set(y_i); \\ \ldots \\ OP_N : IF(x_N), Delay(\tau_m), set/reset(y_N). \end{cases} \quad (3)$$

where: $IF(x_i), \forall i = \overline{1,N}$ - the condition for timeframe τ_j calculation;

$Delay(\tau_j)$ - delays of the enabled transition firing;

$set/reset(y_i)$ - set or reset of the control signal after a timeframe τ_j. In case of complex processes one control signal y_i can have more then one timeframe $\{\tau_j, \tau_{j+1}, \ldots\}$ that synchronize control signal transition from one state to another.

VII. EXAMPLE OF A CONTROL SYSTEM SYNTHESIS

Let's have a RTP that contains four parallel operations, described by the following model:

$$RTP = \begin{cases} OP_1 : IF(x_1 = 1), y_1 = 0, Delay(1ms), \\ y_1 = 1, Delay(1ms), y_1 = 0; \\ OP_2 : IF(x_2 = 1), y_2 = 1, \\ Delay(2ms), y_2 = 0; \\ OP_3 : IF(x_3 = 1), y_3 = 1, \\ Delay(3ms), y_3 = 0; \\ OP_4 : IF(x_4 = 1), Delay(2ms), \\ y_4 = 1, Delay(1ms), y_4 = 0, \\ Delay(1ms), y_4 = 1, \\ Delay(2ms), y_4 = 0. \end{cases} \quad (4)$$

Model (4) has the following time diagram (Figure 4):

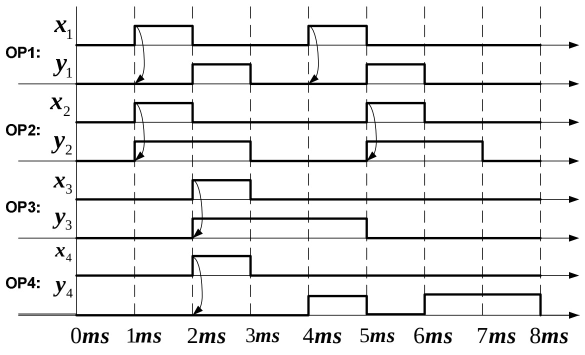

Fig. 4. Time diagram of model (4).

Control signals for operations OP_1, OP_2, OP_3 have only one timeframe; control signal for operation OP_4 has two timeframes. This fact proves the ability of HTPN to generate complex control signal with programmed length.

The HTPN for control system of the process defined by model (4) is presented in Figure 5.

HTPN consists of two subnets: HPN TC – for generation of timeframes $\theta = \{1, 1, 2, 3, 2, 1, 1, 2\} ms$ and HPN PC – for control signals $Y = \{y_1, y_2, y_3, y_4\}$ generation according to input signals $X = \{x_1, x_2, x_3, x_4\}$.

The specification of the HTPN components: $x_1, \ldots, x_4$ - input variables to the control system; $y_1, \ldots, y_4$ - control variables; RTC - synchronization signal for timeframes $\theta = \{1, 1, 2, 3, 2, 1, 1, 2\} ms$ generation (a common $1ms$ divider can be defined for timeframes, it will determines the 1.0KHz generator frequency); $t1, t2, t3, t4, t14, t16, t18, t20$ - transitions for synchronization of timeframes generation; $Delay(\tau_i)$ - places for clock pulses counting necessary for timeframes generation; $D1,\ldots,D7$ - intermediate places for control signals generation;

$t5,t6,t7,t8,t9,t10,t11,t12,t13,t17,t19,t21$ - transitions for control signals generation.

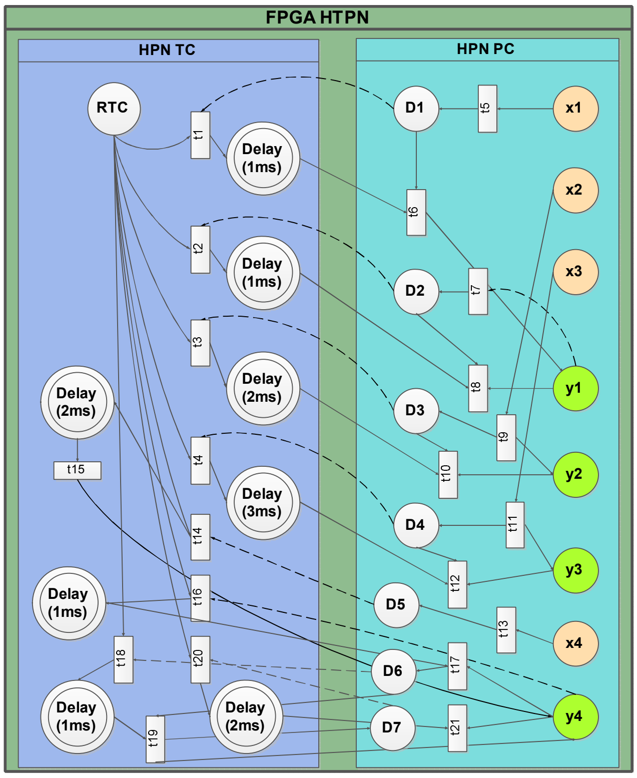

Fig. 5. HTPN for model (4).

VIII. CONCLUSION

In this paper an approach for real-time control systems synthesis based on Petri nets models has been presented. For behavioral specification and modeling of a control system, Timed Synchronous Petri nets (TSPN) has been developed. TSPN is used to analyze functional properties of the designed control system and time constraints. Hard Timed Petri nets (HTPN) was proposed to perform the conversion of the model to logic circuit. HTPN consists of functional elements and logical connections between them. The direct implementation of the HTPN model into FPGA circuit ensures that the behavioral properties and time constraints, under which activate the control system, will be respected. In order to validate the proposed method an example of control system synthesis has been presented.

ACKNOWLEDGMENTS

The researchers conducted in this paper are part: of the 11.817.08.55A project "Advanced methods and technologies for design and development of reconfigurable computing applications", and of the 14.820.18.02.03/U Moldova - Ukraine bilateral project "Distributed sensors network with reconfigurable computing nodes".

REFERENCES

[1] Baer J.L. Microprocessor Architecture, From simple pipelines to chip multiprocessors, Cambridge University Press, ISBN-13978-0-521-76992-1, 2010.
[2] Bondalapti K. and Prasanna V. Reconfigurable computing systems. Proceedings of the IEEE, Vol. 90, No. 7, July, 2002.
[3] Buell D., El-Ghazawi T., Gaj K. and Kindratenko,V. High-Performance reconfigurable computing. IEEE Computer Society, March, 2007.
[4] Cassez F. and Roux O.H. Structural translation from time Petri nets to timed automata. Journal of Systems and Software, 79(10):1456–1468, 2006.
[5] Cristea V. Algoritmi de prelucrare paralela. Matrix Rom, Bucuresti, 2002.
[6] El-Ghazawi T., El-Araby E., Miaoqing Huang, Gaj K., Kindratenko V. and Buell D. The promise of high-performance reconfigurable computing. IEEE computer society, February, 2008, pp. 69 -76.
[7] Lime D. and Roux O.H. Model checking of time Petri nets using the state class timed automaton. Journal of Discrete Events Dynamic Systems - Theory and Applications (DEDS),16(2):179–205, 2006.
[8] Peterson J.L. Petri Net Theory and the Modeling of Systems, Prentice- Hall, 1981.
[9] Sudacevschi V. Teză de doctor în tehnică „Sinteza structurilor de procesare concurentă a datelor" UTM, Chişinău, 2009, 165 p.
[10] Todman T.J., Constantinides G.A., Wilton S.J.E, Luk W. and Cheung P.Y.K. Reconfigurable computing: architectures and design methods. IEEE Proceedings of Computer Digital Technologies, Vol. 152, No. 2, March, 2005.
[11] * * * http://www.altera.com/, accessed 10.09.2014.