Reconfigurable Control Systems Modeling and Design Based on Petri Nets

Viorica SUDACEVSCHI[1], Victor ABABII[2], Valentin NEGURA
Technical University of Moldova
Str. Stefan cel Mare, 168, Chisinau
Republic of Moldova
[1] svm@mail.utm.md, [2] avv@mail.utm.md

Abstract—This paper describes a Discrete Petri net based method for specification, analysis and synthesis of the reconfigurable control system. The control system synthesis is based on Hardware Petri nets that are composed of two kinds of processing elements (Places and Transitions) and data flow path between them. The use of Hardware Petri nets allows the direct translation of the model into FPGA circuits, that substantially reduces the design time and cost. A design example of a control system synthesis for a complex technological process illustrates the proposed method.

Index Terms—Petri Net models, Modeling and Design, Reconfigurable Control Systems, FPGA, AHDL.

I. INTRODUCTION

The deeper understanding of the power of the use of reconfigurable technology platforms made reconfigurable computing one of the most attractive and promising field of research investigation. Their adaptable architecture allows the implementation of the optimal computer systems with low cost and best performance parameters [1, 2]. The application possibilities of reconfigurable architectures are very large: from elementary operations control [3] to complex technological and production processes management [4]. However, reconfigurable architectures designs development needs new approaches in automation of implementation process that will strongly integrate all design steps, start with parametrical and logical modelling of the technological process, modelling of the control system interaction with the controlled process and finish with the final implementation of the computer system.

During last years the importance of Petri nets as a modelling formalism has greatly increased. Various types of Petri nets were proposed to model digital systems, either by using restrictions to the basic formalism, or by adding extensions to it. Several researchers have shown that Hybrid Petri nets are a powerful mechanism to model the behaviour of technological processes. Hybrid Petri nets provide a high quality parametrical and logical modelling [5]. For computer systems modelling and validation Discrete Petri nets are used [6] that are able to support concurrency and synchronisation requirements in system architecture, providing a high functionality of the modelled system.

The automation of the modeling and synthesis process of reconfigurable control system alows the acceleration of the implementation of this system. Such methods were described in [7], where different CAD tools for specification, analysis and synthesis of digital systems were proposed.

In the paper is presented a method for synthesis of the reconfigurable control system for complex technological process. The specification of the control system is done using Discrete Petri net model. The implementation of the control system is based on direct translation of the model into reconfigurable computer arhitecture.

II. THE PETRI NET MODEL OVERVIEW

A *Petri net (PN)* is a 4-tuple $(\boldsymbol{P},\boldsymbol{T},\boldsymbol{F},\boldsymbol{M}^0)$ [8, 9], where:

$\boldsymbol{P}=\{\boldsymbol{p}_1,\boldsymbol{p}_2,...,\boldsymbol{p}_N\}$ is a finite and non-empty set of *places*;

$\boldsymbol{T}=\{\boldsymbol{t}_1,\boldsymbol{t}_2,...,\boldsymbol{t}_L\}$ is a finite and non-empty set of *transitions* ($\boldsymbol{P}\cap\boldsymbol{T}=\varnothing$);

$\boldsymbol{F}\subseteq(\boldsymbol{P}\times\boldsymbol{T})\cup(\boldsymbol{T}\times\boldsymbol{P})$ is a flow relation that defines directed arcs from places to transitions $(\boldsymbol{P}\times\boldsymbol{T})\notin\varnothing$ and transitions to places $(\boldsymbol{T}\times\boldsymbol{P})\notin\varnothing$;

$\boldsymbol{M}^0=\{\boldsymbol{m}_1^0,\ \boldsymbol{m}_2^0,\ ...,\ \boldsymbol{m}_N^0\}$ is the *initial mark places.*

III. HARDWARE IMPLEMENTATION OF PETRI NETS

The computer-based synthesis of the reconfigurable control system from Petri net level to logic design level request the adaptation of the Petri net model to its hardware implemented model. The control system model is considered as a set of processing elements with data flow path between them. The corresponding Petri net model contains two kinds of processing elements $\boldsymbol{P_i}, \boldsymbol{i}=\overline{1,\boldsymbol{N}}$ and $\boldsymbol{T_j}, \boldsymbol{j}=\overline{1,\boldsymbol{L}}$. The arcs between them represent the data flow paths. The arc from processing element $\boldsymbol{P_i}$ to processing element $\boldsymbol{T_j}$ is denoted as $\boldsymbol{P_i}$ **a** $\boldsymbol{T_j}$ and the arc from processing element $\boldsymbol{T_j}$ to processing element $\boldsymbol{P_i}$ is denoted as $\boldsymbol{T_j}$ **a** $\boldsymbol{P_i}$. For hardware implemented *PN* model the data flow depends on the topology of the net.

A ***Hardware Petri Net (HPN),*** is defined as reunion between sets of processing elements and data flows:

$\boldsymbol{HPN}=\boldsymbol{T}\cup\boldsymbol{P}\cup\boldsymbol{P}^{In}\cup\boldsymbol{P}^{Out}\cup\boldsymbol{A}\cup\boldsymbol{M}^0\cup\boldsymbol{M}^{max}$,.

$\boldsymbol{A}=\boldsymbol{A}^+\cup\boldsymbol{A}^-\cup\boldsymbol{A}^S\cup\boldsymbol{A}^T\cup\boldsymbol{A}^I$.

Where:

$T=\{T_1,T_2,...,T_L\}$, $T\neq\varnothing$ is a set of processing elements ***Transition***;

$P=\{P_1,P_2,...,P_N\}$, $P\neq\varnothing$ is a set of processing elements ***Place***;

$A^+=\{A_i^+,\ i=\overline{1,N}\}$, $A^+\neq\varnothing$ is a set of arcs $T_j \overset{+}{\mathbf{a}} P_i$ that represent the condition when the number of tokens in the place is increased and is defined as follows:

$$A_i^+=\begin{cases} a_{ji}^+=1 \mid T_j \overset{+}{\mathbf{a}} P_i, & i=\overline{1,N},\ j=\overline{1,L};\\ a_{ji}^+=0, \quad \textbf{\textit{otherwise}}. \end{cases}$$

$A^-=\{A_i^-,\ i=\overline{1,N}\}$, $A^-\neq\varnothing$ is a set of arcs $T_j \overset{-}{\mathbf{a}} P_i$ that represent the condition when the number of tokens in the place is decreased and is defined as follows:

$$A_i^-=\begin{cases} a_{ji}^-=1 \mid T_j \overset{-}{\mathbf{a}} P_i, & i=\overline{1,N},\ j=\overline{1,L};\\ a_{ji}^-=0, \quad \textbf{\textit{otherwise}}. \end{cases}$$

$A^S=\{A_j^S,\ j=\overline{1,L}\}$, $A^S\neq\varnothing$ is a set of *state arcs* $P_i \overset{S}{\mathbf{a}} T_j$ that determine the enable firing condition of the transition T_j related to the marking of the place P_i the arc is connected with. This set is defined as follows:

$$A_j^S=\begin{cases} a_{ij}^S=1 \mid P_i \overset{S}{\mathbf{a}} T_j, & i=\overline{1,N},\ j=\overline{1,L};\\ a_{ij}^S=0, \quad \textbf{\textit{otherwise}}. \end{cases}$$

State arc connects an input place to a transition and has the ability to check whether a place has a token. The presence of a state arc connecting an input place to a transition means that the transition is only enabled if the input place has a token. The firing change the marking in the enabling arc connected place.

$A^T=\{A_j^T,\ j=\overline{1,L}\}$, is a set of *test arcs*, which has the same function as the set of state arcs, but the firing does not change the marking in the *test arc* connected place.

$$A_j^T=\begin{cases} a_{ij}^T=1 \mid P_i \overset{T}{\mathbf{a}} T_j, & i=\overline{1,N},\ j=\overline{1,L};\\ a_{ij}^T=0, \quad \textbf{\textit{otherwise}}. \end{cases}$$

$A^I=\{A_j^I,\ j=\overline{1,L}\}$, is a set of *inhibitor arcs*, which provides an enabling function, when the place stores no tokens. It is defined as follows:

$$A_j^I=\begin{cases} a_{ij}^I=1 \mid P_i \overset{I}{\mathbf{a}} T_j, & i=\overline{1,N},\ j=\overline{1,L};\\ a_{ij}^I=0, \quad \textbf{\textit{otherwise}}. \end{cases}$$

Inhibitor arc connects an input place to a transition and has the ability to test whether a place is empty. The presence of an inhibitor arc connecting an input place to a transition means that the transition is enabled if the input place has no token. The firing does not change the marking in the inhibitor arc connected place.

$P^{In}\in P$ is a set of processing elements P that represent the input signals connections. (This signal is connected to the FPGA circuit pine and determines the state of the controlled object. If $P_i^{In}=1$ then the place P_i^{In} has the external increment input);

$P^{Out}\in P$ is a set of processing elements P_j that represent the output signals connections (This signal is connected to the FPGA circuit pine and determines the action on the controlled object. If $P_i^{Out}=1$ then the place P_i^{Out} represent the active control signal);

M^0 is the initial mark place;

$M^{\max}$ is the maximal mark place.

The interaction of the control unit, represented by ***HPN*** and the external system is done through input and output signals that are given by processing elements P^{In}, P^{Out} (figure 1).

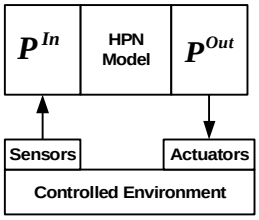

Figure 1. The interaction of the ***HPN*** model with external system (Controlled environment)

IV. PROCESSING ELEMENTS

The processing element T prepares the data processing operation. After analyzing of the global state $S^k=\{(m_i,P_i),\forall i=\overline{1,I}\}$ at the step k of data processing, the condition for step $k+1$ of data processing operation is formed.

The behavior of the processing element T may be described as follows:

$$T_j=\prod\left(\prod_{i=1}^{N_j^S}(A^s),\prod_{i=1}^{N_j^T}(A^T),\prod_{i=1}^{N_j^I}(A^I)\right).$$

The AHDL code for the processing element T is shown in figure 2. The processing element contains 4 inputs.

```
SUBDESIGN T_OBJ
(
        Tin0, Tin1, Tin2, Tin3 : INPUT;
        Clk  : INPUT;
        Set  : INPUT ;
        Reset: INPUT;
        Tout : OUTPUT;
```

```
)
VARIABLE
        T_DFF : JKFF;
        T : NODE;
BEGIN
        T_DFF.Clk = Clk;
        T_DFF.PRN = Set;
        T_DFF.CLRN = Reset;
        T = Tin0 & Tin1 & Tin2 & Tin3;
        T_DFF.J = T;
        T_DFF.K = !T;
        Tout = T_DFF.Q;
END;
```

Figure 2. The AHDL code of the processing element T

The processing element $\boldsymbol{P}$ stores the state value and performs the increment and decrement operation of the number of tokens. The increment operation occurs when one of the input transitions of the processing element $\boldsymbol{P}$ fires. The decrement operation occurs when one of the output transition of the processing element $\boldsymbol{P}$ fires. The number of tokens in $\boldsymbol{P}$ at the step $k+1$ of data processing, denoted by m_i^{k+1}, is changed according to the following rules:

$$m_i^{k+1} = \begin{cases} m_i^k + 1 \;\Big|\; \sum_{j=1}^{L_i^+}(A^+_{ij}) = 1,\ \forall\ m_i^k = 0; \\ m_i^k - 1 \;\Big|\; \sum_{j=1}^{L_i^-}(A^-_{ij}) = 1\ \forall\, m_i^k = 1; \\ m_i^k \;\Big|\; \sum_{j=1}^{L_i^+}(A^+_{ij}) = 0\ \&\ \sum_{j=1}^{L_i^-}(A^-_{ij}) = 0; \\ m_i^k \;\Big|\; \sum_{j=1}^{L_i^+}(A^+_{ij}) = 1\ \&\ \sum_{j=1}^{L_i^-}(A^-_{ij}) = 1; \end{cases} , i = \overline{1, N};$$

where: m_i^k is the number of tokens in $\boldsymbol{P_i}$ at the step k of data processing, L_i^+ and L_i^- are the total number of increment and decrement arcs to the place $\boldsymbol{P_i}$.

The AHDL code of processing element P is presented in figure 3. The elaborated processing element contains 4 increment inputs and 4 decrement inputs.

```
SUBDESIGN P_OBJ
(
        Pinc0, Pinc1, Pinc2, Pinc3 : INPUT;
        Pdec0, Pdec1, Pdec2, Pdec3 : INPUT;
        Clk         : INPUT;
        Set         : INPUT;
        Reset : INPUT;
        Pout : OUTPUT;
)
VARIABLE
        P_DFF : JKFF;
        INC : NODE;
        DEC : NODE;
BEGIN
        P_DFF.CLK = Clk;
        P_DFF.PRN = Set;
        P_DFF.CLRN = Reset;
        INC = (Pinc0 # Pinc1 # Pinc2 # Pinc3);
        DEC = (Pdec0 # Pdec1 # Pdec2 # Pdec3);
        IF (INC & DEC)  OR  (!INC & !DEC)  THEN
         P_DFF.J = P_DFF.Q;
         P_DFF.K = !P_DFF.Q;
        ELSE P_DFF.J = INC;
         P_DFF.K = DEC;
```

Figura 3. The AHDL code of the processing element P

V. DESIGN EXAMPLE

In order to illustrate the proposed method, a technological control system is used as example. The technological process (figure 4) consists in receiving the mixture that is composed of three chemical components X_1, X_2 and X_3 from three reservoirs V_1, V_2 and V_3. The result mixture X_4 is placed in reservoir V_4. For chemical components dosage the reservoirs have high level identification sensors (s_1, s_3 and s_5) and low level identification sensors (s_2, s_4 and s_6). Sensor s_7 semnalise the end of evacuation process of the result mixture from reservoir V_4. Signals u_1, u_3 and u_5 are used to control the reservoirs filling. Signals u_2, u_4 and u_6 are used to control the reservoirs discharge. Signal u_7 control the mixture discharge from reservoir V_4. The acceleration of the mixture preparation is controled by signal u_8.

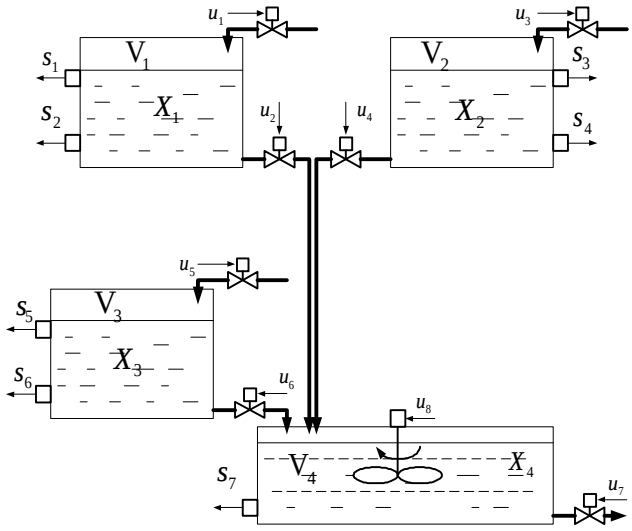

Figure 4. Technological process

Petri net model for described technological process is presented in figure 5. This model is used for modeling and control of the technological process as it specifies the main functions of the technological process and contains necessary structures to generate control signals.

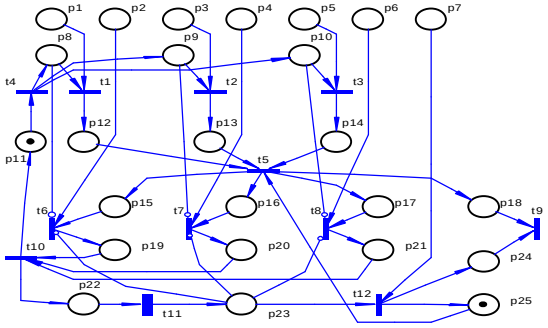

Figure 5. Petri net model

Connection of input signals to control sistem HPN is done in the following way: $S_1 \to P_1$, $S_2 \to P_2$, $S_3 \to P_3$, $S_4 \to P_4$, $S_5 \to P_5$, $S_6 \to P_6$, $S_7 \to P_7$.

Connection of output signals that have the function to control the technological process is done in the following way: $P_8 \to U_1$, $P_9 \to U_3$, $P_{10} \to U_5$, $P_{15} \to U_2$, $P_{16} \to U_4$, $P_{17} \to U_6$, $P_{18} \to U_8$, $P_{23} \to U_7$.

VI. CONTROL SYSTEM MODELLING AND IMPLEMENTATION RESULTS

Petri net modelling and validation was performed using a specialised program VPNP [6]. As a result, incidence matrixes A^+, A^-, A^S, A^T, A^I were obtained. The conversion of incidence matrixes to AHDL code was performed using a compiler program, elaborated by authors.

Control system configuration code for implementation in FPGA circuits was obtained using a specialized tool MAX+Plus II , Altera [1]. The implementation results the logical symbol of the obtained control system is presented in figure 6.

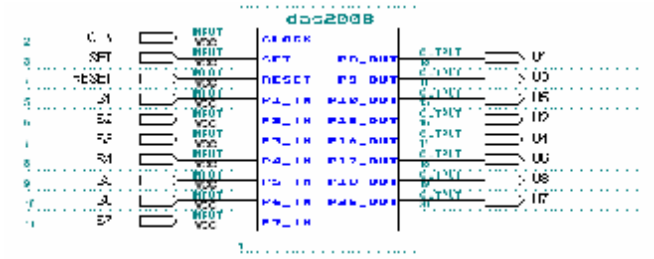

Figure 6. Logical symbol of the control system

The simulation results are presented in figure 7, where:

Time = 10 ns – start the filling up process of *V1*, *V2* and *V3* reservoirs;

Time = 50 ns – high level identification sensors S_1, S_3 and S_5 are active;

Time = 100 ns – end the filling up process of *V1*, *V2* and *V3* reservoirs;

Time = 120 ns – start the evacuation process from *V1*, *V2* and *V3* reservoirs ; start the mix up process in *V4* reservoir;

Time = 200 ns – *V1*, *V2* and *V3* reservoirs are empty;

Time = 240 ns – stop the evacuation process from *V1*, *V2* and *V3* reservoirs;

Time = 280 ns – start the evacuation process from *V4* reservoir and fill up process in *V1*, *V2* and *V3* reservoirs;

Time = 370 ns – *V4* reservoir is empty;

Time = 420 ns – stop the evacuation process from *V4* reservoir;

Time = 440 ns – stop the mix up process.

The mentioned operations are repeated for each production cycle.

VII. CONCLUSIONS

In this paper an approach of reconfigurable computer systems modeling and design based on Hardware Petri net models has been presented. The main research direction has been pointed on real time control of technological processes. The use of Hardware Petri nets in CAD tools allows the automation of the FPGA implementation process and substantially reduces the design time and efforts. As an example was choused a complex technological process with several parallel operations that requests a concurrent data processing. For synthesis of the control system a direct translation method of the Hardware Petri net model into AHDL code was used. The configuration code for implementation of the control system into FPGA circuits was obtained using a specialized tool MAX+Plus II.

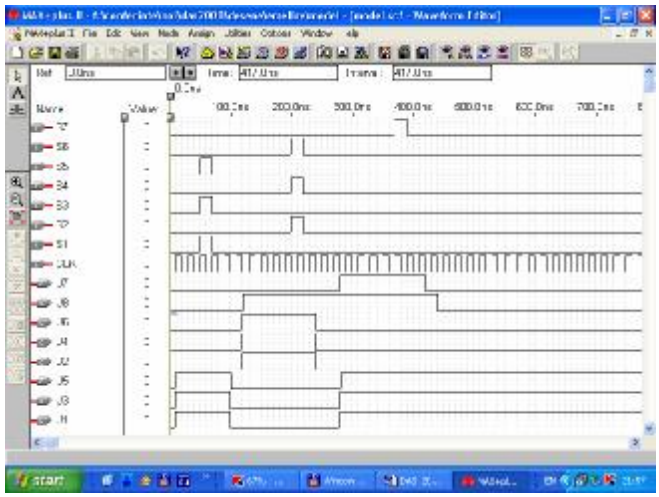

Figure 7. The simulation results

REFERENCES

[1] http://www.altera.com/.
[2] http://www.xilinx.com/.
[3] K. Leijten-Nowak, "Reconfigurable electronic device with dual port memory mode", PHNL020827, Invention Disclosure ID609469 ("Distributed Dual-Port Memory Architecture for Reconfigurable Logic Devices"), February 2002.
[4] Manfredi Bruccoleri. Reconfigurable control of robotized manufacturing cells Robotics and Computer-Integrated Manufacturing. Volume 23 , Issue 1 (February 2007). Pages 94-106. ISSN:0736-5845.
[5] Wikarski, D. Petri Net Tools: a Comparative Study, ISST-Bericht 39/96 of the Fraunhofer-Gesellschaft e.g., Berlin, 1996
[6] E. Gutuleac, C. Bosneaga, A. Reilean. (2002) VPNP – Software Tool for Modeling and Performance Evaluation Using Generalized Stochastic Petri Nets. 6th International Conference on DAS – 2002, Suceava, Romania, May 23-25.
[7] Publicatia din Oradia 2005.
[8] Peterson, J. Petri Net theory and modeling of systems. New York, 1984.
[9] T. Murata Petri Nets: properties, analysis and applications Proceedings of IEEE, vol. 77, pp. 541-580, Apr. 1989.