UDC 681.32

V.V. ABABII, V.M. SUDACEVSCHI

Technical University of Moldova, Moldova

THE MODELLING AND DESIGN OF RECONFIGURABLE CONTROL SYSTEMS

Design of real-time control systems requires new methodologies in their modelling, verification, validation and implementation. In the paper a complex integrated design system is presented. The design process starts with analysing of the Petri net model of the control system in a special software environment that performs modelling, verification, validation and performance evaluation of the model, its conversion into AHDL code (Hard Petri net), simulation of the obtained code in MAX+ Plus II environment and FPGA or CPLD configuration of the control system.

AHDL Design, Petri Net Models, Hardware Implementation, Control System

Introduction

The increasing complexity of real-time control systems requires new approaches for their modelling, synthesis and verification. A lot of scientific research studies propose different methods for design stages integration into an automatic flow with minimal human participation [1, 2, 3, 4]. These methods are based on Petri net type models as the first step in control system design and their conversion into a program code that is executed on PLC systems. Other research direction is the Petri net model implementation in basic logic elements that can be used in control systems or in modelling systems [5, 6, 7].

In this paper is presented an integrated design environment that support synthesis, modelling and validation of a control system with concurrent data processing based on Petri net model, performance analysis and translation of this model into AHDL code that allows control system configuration into FPGA or CPLD circuits.

Diagram of synthesis flow

Control system synthesis is executed according to the diagram that is presented in fig. 1.

Synthesis stages description:

PN Models Source – Petri net model that is proposed for analysing is introduced in graphical form;

VPNP Tool – software tool that allows inserting and modifying in an interactive mode the Petri net model;

PN Models Library – the library with Petri net models;

Analysis (Reachability graph & Structural analysis) – The proposed Petri net model is analysed in order to determine the set of reachable states and to form the reachability graph. The structural analysis determines the main properties of the model such as its safety and viability;

MI and MO generation – incidence matrix and initial marking generation and their storage in corresponding files;

HDL Compiler – AHDL code compilation based on matrixes **.imf* and **.rag*;

HDL Objects Library – the library with standard AHDL objects that are used to form AHDL code of a Petri net;

HDL code – the obtained after compilation AHDL code;

Max Plus + II Design Tool – MAX+PLUS II software is a fully integrated, architecture-independent package for designing logic with ALTERA programmable logic devices;

FPGA or CPLD Device – FPGA or CPLD configuration of the Petri net model.

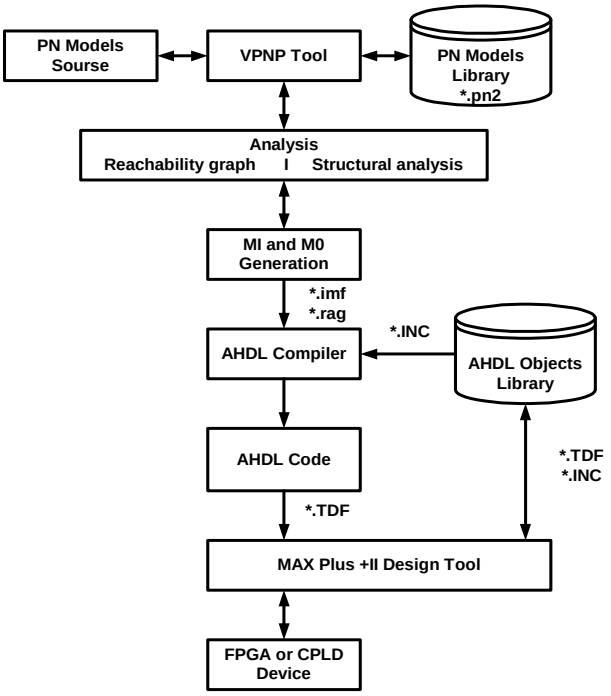

Fig. 1. Diagram of synthesis flow

The VPNP Tools

The interactive environment VPNP represents a software tool with a graphical interface, designed for Petri net models analysing. It allows to draw a graphical Petri net model, to store into a file and to read from a file these models and to perform the structural analysis of the models with visualization of the results [8]. After analysing of the Petri net model the incidence matrix IM (**.imf*) and initial state (**.rag*) files are obtained.

Petri net model for hardware implementation

A Petri net *(PN)* is a 5-tuple:

$$PN = \left(P, T, W(p,t), W(t,p), M_0\right),$$

where: $P = \left\{p_i, \forall i = \overline{1,I}\right\}$ – is a finite and non-empty set of places;

$T = \left\{t_j, \forall j = \overline{1,J}\right\}$ – is a finite and non empty set of transitions;

$W(p,t) = \left\{\left(p_i, t_j\right),\ \forall\ i = \overline{1,I},\ j = \overline{1,J}\right\}$ – is a set of arcs from place P to transition T;

$W(t,p) = \left\{\left(t_j, p_i\right),\ \forall\ j = \overline{1,J},\ i = \overline{1,I}\right\}$ – is a set of arcs from transition T to place P;

$M_0 = \left\{m_1, m_2, ..., m_I\right\}$ – initial marking.

The Petri net changes its states according to functional rules that are defined for each class of Petri nets [9, 10, 11].

The architecture of the system with concurrent data processing represents a set of processor elements with data flow interconnections [12, 13]. For a Petri net model data flow will depend on the internal structure of the net model. Taking this into consideration, we can define a Hardware Petri Net (*HPN*) as a set of processor elements (transitions and places) and data flows (arc connections):

$$RPH = \mathrm{T} \cup \mathrm{P} \cup \mathrm{A}^+ \cup \mathrm{A}^- \cup A^S \cup \mathrm{P}^I \cup \mathrm{P}^O,$$

where: $T = \{t_1, ..., t_J\}$ – is set of transition type processor elements; $P = \{p_1, ..., p_I\}$ – is a set of place type processor elements; $\mathrm{A}^+ = \left\{A_1^+, A_2^+, ..., A_J^+\right\}$ – is a set of increment connections to each position, where:

$$A_j^+ = \begin{cases} a_{j,i}^+ = 1, \text{if exists a connection between } T \text{ and } P; \\ a_{j,i}^+ = 0, \text{if do not exist a connection between } T \text{ and } P; \end{cases}$$

$\mathrm{A}^- = \left\{A_1^-, A_2^-, ..., A_I^-\right\}$ – is a set of decrement connections from each position, where:

$$A_i^- = \begin{cases} a_{i,j}^- = 1, \text{if exists a connection between } P \text{ and } T; \\ a_{i,j}^- = 0, \text{if do not exist a connection between } P \text{ and } T. \end{cases}$$

Incidence matrix is obtained as: $IM = A^+ \cup A^-$.

$\mathrm{A}^S = \left\{A_j^S,\ j = \overline{1,L}\right\}$, $\mathrm{A}^S \neq \varnothing$ – is a set of state connections, where $A^S = \left(A^-\right)^T$;

$P^{In} = \left\{P_j^{In},\ j = \overline{1,L^I}\right\}$ – is a set of place type processor elements, also are certain as input signals of a condition of operated object, where $P^{In} \in P$;

$P^{Out} = \left\{P_j^{Out},\ j = \overline{1,L^O}\right\}$ – is a set of place type processor elements, also are certain as output signals of control by object, where $P^{Out} \in P$.

The pair $\left(m_i, P_i\right)$ determines the state of the proces-

sor element P_i. The set of all states for places $S^k = \{(m_i, P_i),\ \forall\ i = \overline{1,I}\}$ determines the global state of the system at k iteration, k where $k \in K$. The state $S = \bigcup_{1}^{K} S^k$ determines the set of allowed states for the system and the reachability graph for the Petri net model.

Processor elements specification

The hardware implementation of Petri Net contains two main parts: the processor element transition (T) and the processor element place (P).

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

The logic symbol of a functional element transition is presented in fig. 2, where:

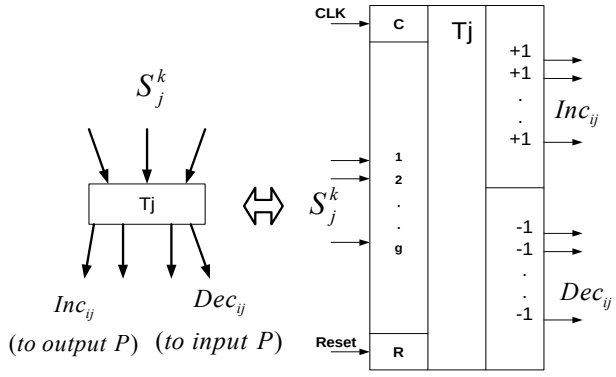

Fig. 2. Functional element Tranzition

$CLK(C)$ – clock signal; Inc_{ij} – increment outputs connected to all output places to this transition; Dec_{ij} – decrement outputs connected to all input places to this transition, S_j^k – Petri net model state signal inputs for transition T_j. For all transitions the logic function $Inc_{ij} = Dec_{ij} = \prod_{m=1}^{M} \left(s^k_{j_m} \right)$ is formed that allows the transition to fire only if all inputs $m = 1,...,M$ will have the logic value "1".

The processor element Place P stores the state value and performs the increment and decrement operation of the number of markers. The logic symbol for processor element Place P is shown in fig. 3, where:

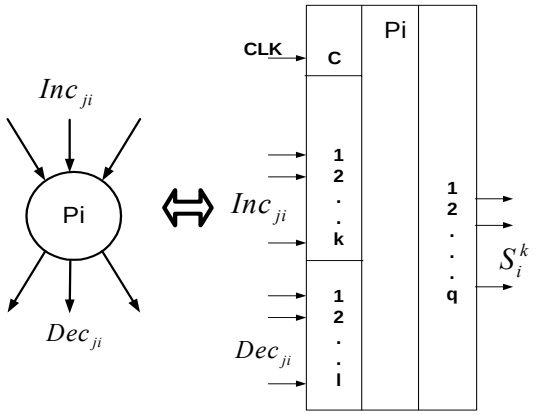

Fig. 3. Functional element Place

$CLK(C)$ – clock signal; Inc_{ji} – enable inputs for increment operation of the number of markers in place; Dec_{ji} – enable inputs for decrement operation of the number of markers in place; S_i^k – place state at the k iteration step that determines the marking presence in place.

The number of markers in place m_i^{k+1} is changed according to the following formula:

$$m_i^{k+1} = \begin{cases} 1 \ \Big| \sum_{j=1}^{J} (Inc_{ji}) = 1 \ \ \& \ \ (m_i^k = 1 \ \vee \ m_i^k = 0); \\ 0 \ \Big| \sum_{j=1}^{J} (Dec_{ji}) = 1 \ \ \& \ \ m_i^k = 1; \\ m_i^k \ \Big| \sum_{j=1}^{J} (Inc_{ji}) = 0 \ \ \& \ \ \sum_{j=1}^{J} (Dec_{ji}) = 0; \\ m_i^k \ \Big| \sum_{j=1}^{J} (Inc_{ji}) = 1 \ \ \& \ \ \sum_{j=1}^{J} (Dec_{ji}) = 1; \end{cases}$$

$\forall i = \overline{1,I}.$

HDL Compiler

A HDL compiler performs conversion of the Petri net model that is defined by the incidence matrix IM (file *.imf) and initial state matrix M0 (file *.rag), obtained in VPNP software tool, to AHDL code.

The dialog window of the software tool HDLCS allows to insert the incidence matrix IM (command ***OPEN***), the initial marking M0 (command ***LOAD M0***)

and to save the AHDL code of the Petri net model. Command ***PROCESS*** starts the compilation. The AHDL code is obtained after processor elements selection from HDL Library according to their characteristics, their interconnections according to the incidence matrix IM and generation of the file that contains the source AHDL code with TDF extension.

At the first step of AHDL code generation in the file are included processor elements Place and Transition, the global synchronization clock is defined, and interconnections between processor elements are formed.

Conclusions

We have described the design of a control system using Petri nets. The proposed integrated system uses Petri net model for modelling, verification and validation of control system functionality, conversion of the model into HDL code and its implementation into FPGA or CPLD circuits. The proposed method allows a high flexibility in quick reconfiguration of control algorithms. The obtained results prove the reliability of the integrated system. The authors plan to continue investigation of this method. One of the most important research directions is the concurrent data processing analysis, new synchronization methods for data processing operations, functional extension of the processor elements for Timed Petri net implementation.

References

1. Fernandes J.M., Proeneca A.J.; Adamski M.A. VHDL Generation from Petri Net Parallel Controller Specification // Proceeding of EUROVHDL' 95, Brighton, GB, 18. – 22.9.1995.

2. Fengler W., Wendt A., Adamski M.A., Monteiro J.L. M. P.: Petri Net Based Program Design for Controller Systems // Proceeding of13th IFAC World Congress, San Francisco, June 30 – July 5, 1996.

3. Jones A.H., Uzam M., Khan A.H., Karimzadgan D., Kenway S.B. A General Methodology for Converting Petri Nets Into Ladder Logic: The TPLL Methodology // Proceedings of the 5th International Conference on Computer Integrated Manufacturing and Automation Technology – CIMAT'96. – May 1996. – France. – P. 357-362.

4. Jones A.H., Uzam M., Ajlouni N. Design of Discrete Event Control Systems for Programmable Logic Controllers Using T-timed Petri Nets // Proc. of the 1996 IEEE Int. Symp. on Computer-Aided Control System Design – CACSD'96. – Dearborn, MI, USA. – September 15-18, 1996. – P. 212-217.

5. Uzam M., Avci M., Kürsat Yalçin M. Digital Hardware Implementation of Petri Net Based Specifications: Direct Translation from Safe Automation Petri Nets to Circuit Elements // The International Workshop on Discrete-Event System Design, DESDes'01. – Przytok near Zielona Gora, Poland.– June 27÷29, 2001.

6. Bundell G.A. An FPGA implementation of the Petri Net firing algorithm // In Proc. 4th Australasian Conf. on Parallel and Real-Time Systems. – 1997. – P. 434-445.

7. Morris J. et al. A Re-configurable Processor for Petri Net Simulation // Proceedings of the 33rd Hawaii International Conference on System Sciences. – 2000.

8. Guţuleac E., Reilean A., Boşneaga C. Visual Petri Net plus –integrate Program Package for modeling using Stochastic Petri Nets. // Proc. of the 3-rd Int. Conf. on "Information Technologies-2001", BIT+2001, 11-13 April, Chişinău, Moldova. – 2001. – Vol. 1. – P. 46.

9. Peterson J. Petri Net theory and the modelling of systems. – New-York, 1984.

10. Murata T. Petri Nets: Properties, Analysis and Applications // Proceeding of the IEEE. – 1989. – Vol. 77, no. 4. – P. 541-580.

11. Gutuleac E. Modelarea şi evaluarea performanţelor sistemelor de calcul prin reţele Petri. Partea I şi II, DEP UTM. – Chişinău. –1998.

12. Culler D. Singh J.P. Parallel Computer Architecture. – Morgan Kaufmann, ISBN 0-678-954-341-2, 1999.

13. Jordan H. Fundamentals of parallel processing: Algorithms, architecture, language. – Prentice Hall, ISBN 0-13-901158-7, 2003.

Поступила в редакцию 2.02.2007

Рецензент: д-р техн. наук, проф. А.М. Романкевич, Национальный технический университет Украины «КПИ», Киев.