Proceedings of ICCCC 2006, Băile Felix - Oradea, Romania
pp. 57-61

Control Systems Modelling and Design for Processes Synchronization

Victor Ababii, Viorica Sudacevschi, Emilian Guţuleac

Abstract: Design of real-time control systems requires new methodologies in their modelling, verification 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.

Keywords: HDL Design, Petri Net Models, Hardware Implementation, Performance evolution, System Control.

1 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.

2 Diagram of synthesis flow

Control system synthesis is executed according to the diagram that is presented in Figure 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;

Analysis (**Reachability graph and 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.

3 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 ($*.imf$) and initial state ($*.rag$) files are obtained.

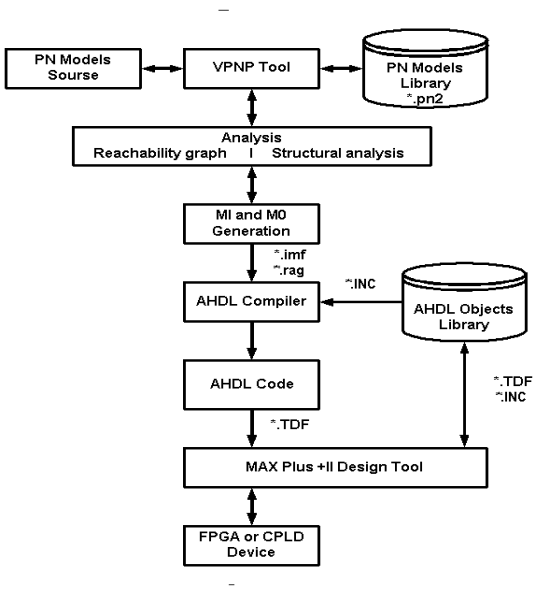

Figure 1: Diagram of synthesis flow

4 Petri net model for hardware implementation

A Petri net(PN) is a 5-tuple:$PN =< P,T,W^+,W^-,M_0 >$,where:

$P = \{p_i, i = \overline{1,I}\}$-is a finite and non-empty set of places; $T = \{t_j, j = \overline{1,J}\}$- is a finite and non empty set of transitions; $W^+ = \{(p_i,t_j)\}$- is a set of arcs from place p_i to transition t_j; $W^- = \{(t_j,p_i)\}$- is a set of arcs from transition t_j to place p_i ; $M_0 = \{m_1,m_2,...,m_I\}$- is the initial marking.

The Petri net changes its states according to functional rules that are defined for each class of Petri nets [13, 14, 3]. The architecture of the system with concurrent data processing represents a set of processor elements with data flow interconnections [12]. 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 ( ) as a set of processor elements (transitions and places) and data flows (arc connections):

$HPN = \{T \cup P \cup A^+ \cup A^-\}$, where:

$T = \{t_1,...,t_J\}$- is set of transition type processor elements;

$P = \{p_1,...,p_J\}$- is a set of place type processor elements;

$A^+ = \{A_1^+,A_2^+,...,A_J^+\}$- is a set of increment connections to each place, where:

$$A_j^+ = \begin{cases} a_{j,i}^+ = 1 & \text{if exists a connection between } t_j \text{ and } p_i, \\ a_{j,i}^+ = 0 & \text{if do not exist a connection between } p_i \text{ and } t_j. \end{cases}$$

$A^- = \{A_1^-,A_2^-,...,A_J^-\}$- is a set of decrement connections from each place, where:

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

Incidence matrix is obtained as: $IM = A^+ \cup A^-$. The pair (m_i,P_i) determines the state of the processor element P_i. The set of all states $S^k = \{(t_j,P_i), \forall i = \overline{1,I}\}$ for places determines the global state of the system at k iteration, where, $k \in K$. The state $S = \cup_{k=1}^{K} S^k$ determines the set of allowed states for the system and the reachability graph for the Petri net model.

5 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 $S^k = \{(t_j, P_i), \forall i = \overline{1,I}\}$ analysing at the step 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 Figure 2a, where: $CLC(C)$- 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 $Inc_{ij} = Dec_{ij} = \prod_{m=1}^{M}(s_{jm}^k)$ 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 tokens. The logic symbol for processor element Place P is shown in Figure 2b, where: $CLC(C)$- clock signal; Inc_{ij} - enable inputs for increment operation of the number of markers in place; Dec_{ij}- enable inputs for decrement operation of the number of markers in place; S_j^k- place state at the k iteration step that determines the marking presence in place. The number of tokens in place $m_i^{k+1}, i = \overline{1,I}$ is changed according to the following formula:

$$m_i^{k+1} = \begin{cases} 1 & \text{if} \quad \sum_j^J (Inc_{ij}) = 1 \wedge (m_i^k) = 1 \vee (m_i^k) = 1 \\ 0 & \text{if} \quad \sum_j^J (Dec_{ij}) = 1 \wedge (m_i^k) = 1 \\ m_i^k & \text{if} \quad \sum_j^J (Inc_{ij}) = 0 \wedge (m_i^k) = 0 \\ m_i^k & \text{if} \quad \sum_j^J (Inc_{ij}) = 1 \wedge (m_i^k) = 1 \end{cases}$$

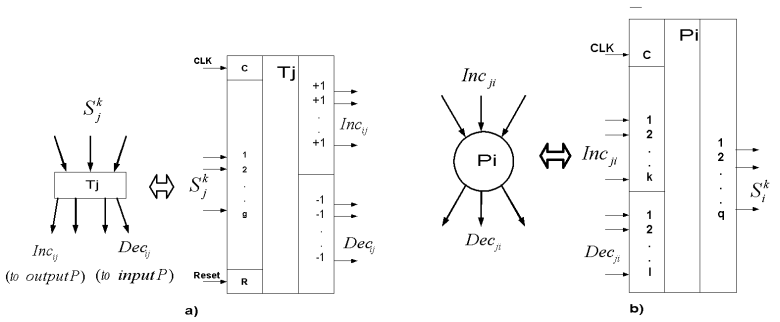

Figure 2: Functional element (a) Transition and (b) Place

6 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.

7 An Example of Control System for Synchronisation of Data Communication

The proposed method was used to design a control unit for data transfer in a computer system. The structure of the control unit is presented in figure 3a. Where the source and destination block communicate using following signals: STB - Strobe when sending byte by Data Bus and ACK signal to confirm the data reception.The Petri net model for control system modelling and control unit implementation is shown in figure 3b and figure 3c,

respectively. After VPNP structural analysis of Petri net model the incidence matrix and initial marking (M0 = [0,0,0,0,1]) where obtained.

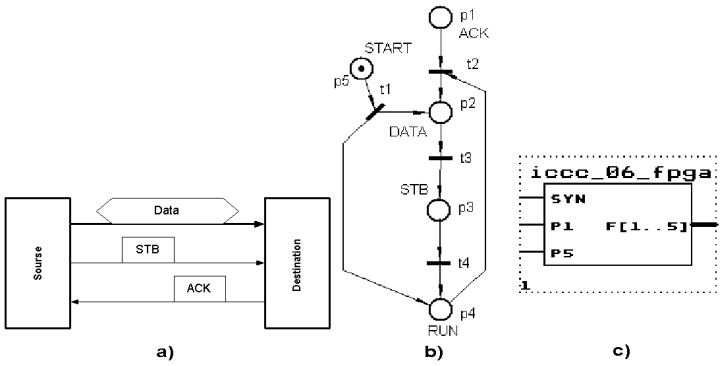

Figure 3: The structure of Control Unit (a) Petri net model for implementation (b) and the Control Unit

Table 1. The AHDL code

```
INCLUDE "place2x1.inc";
INCLUDE "transition1.inc";
INCLUDE "place1x1.inc";
INCLUDE "transition2.inc";
SUBDESIGN iccc_06_fpga
(
        SYN: input;
        p1 : input; %ACK%
        p5 : input; %START%
        f[1..5] : output; %f[3]  -STB%
)
VARIABLE
        p2: place2x1;
        p3: place1x1;
        p4: place2x1;
        t2: transition2;
        t3: transition1;
        t1: transition1;
        t4: transition1;
BEGIN
        t4.Min0=p3.Mout;
        t4.SYN=!SYN;
         t2.Min0=p1; %.Mout; %
        t2.Min1=p4.Mout;
        t2.SYN=!SYN;
         t3.Min0=p2.Mout;
        t3.SYN=!SYN;
        t1.Min0=p5; %.Mout; %
        t1.SYN=!SYN;
        p2.Inc0=t1.Dec_Inc;
        p2.Inc1=t2.Dec_Inc;
        p2.Dec0=t3.Dec_Inc;
        p2.SYN=SYN;
        p2.nCLR=VCC;
        p2.nPR=!(!p5 & GND & !SYN);
                                %Set M0%
        f[2]=p2.Mout;
        p3.Inc0=t3.Dec_Inc;
        p3.Dec0=t4.Dec_Inc;
        p3.SYN=SYN;
        p3.nCLR=VCC;
        p3.nPR=!(!p5 & GND & !SYN);
                                %Set M0%
        f[3]=p3.Mout;
        p4.Inc0=t1.Dec_Inc;
        p4.Inc1=t4.Dec_Inc;
        p4.Dec0=t2.Dec_Inc;
        p4.SYN=SYN;
        p4.nCLR=VCC;
        p4.nPR=!(!p5 & GND & !SYN);
                                %Set M0%
        f[4]=p4.Mout;
END;
```

The AHDL code is presented in table 1. The code is processed in MAX + Plus II tool. The obtained control unit for data transfer is presented in figure 3c, where: SYN - synchronization signal; P1 - ACK input generated by the destination object; P5 -input for data transfer initialisation; F[2] -transfer to data bus; F[3] - generates STB signal for data transfer; F[4] - the system is ready for the next operation. The statistic report obtained after FPGA compilation shows that it was used 12 LCs. The timing diagrams obtained after simulation confirm the correctness of control system functionality.

8 Conclusions

We have described the design of a control system using Petri nets. The proposed integrated system uses Petri net model for modelling and verification 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.

We 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] J. M. Fernandes, A. J. Proeneca,M. A. Adamski, "VHDL Generation from Petri Net Parallel Controller Specification," *ser. In Proceeding of EUROVHDL' 95*, Brighton, GB, 18.-22.09, 1995.

[2] W. Fengler, A.Wendt, M.A. Adamski, J. L. Monteiro, "Net Based Program Design for Controller Systems," *In Proceeding of 13th IFAC World Congress*, San Francisco, June 30 - July 5, 1996.

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

[4] A.H. Jones, M. Uzam, N. Ajlouni, "Design of Discrete Event Control Systems for Programmable Logic Controllers Using T-timed Petri Nets," *In Proceedings Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design - CACSD'96*, Dearborn, MI, USA, September 15 - 18, pp. 212 - 217, 1996 .

[5] Murat Uzam, Mutlu Avci and M. Kursat Yalcin, "Digital Hardware Implementation of Petri Net Based Specifications: Direct Translation from Safe Automation Petri Nets to Circuit Elements," *In Proceedings of the International Workshop on Discrete-Event System Design, DESDes'01*, June 27-29, Przytok near Zielona Gora, Poland, 2001.

[6] G. A. Bundell, "An FPGA implementation of the Petri Net firing algorithm," *In Proceedings of the 4th Australasian Conf. on Parallel and Real-Time Systems*, pp. 434-445, 1997.

[7] John Morris et al., "A Re-configurable Processor for Petri Net Simulation, " *In Proceedings of the 33rd Hawaii International Conference on System Sciences,*2000.

[8] E.Gutuleac, A. Reilean, C. Bosneaga, "Visual Petri Net plus -integrate Program Package for modeling using Stochastic Petri Nets," *Proceedings of the 3-rd International Conference on Information Technologies-2001, BIT+2001*, 11-13 April, Vol.1, Chisinau, Moldova, Page 46, 2001.

[9] J. Peterson, "Petri Net theory and the modeling of systems," New-York, 1984.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications," *In Proceeding of the IEEE, vol. 77, no. 4*, pp. 541-580, 1989.

[11] E. Gutuleac, "Modelarea si evaluarea performantelor sistemelor de calcul prin retele Petri," Partea I, DEP UTM, Chisinau, 1998.

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

Victor Ababii,
Viorica Sudacevschi,
Emilian Guţuleac
Technical University of Moldova
Computer Science Department
Address: 168, Bd. Stefan cel Mare, MD-2004
Chişinău, Republic of Moldova
E-mail: {avv, svm, egutuleac}@mail.utm.md