PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

FROM MODELLING TO IMPLEMENTATION OF CONCURRENT CONTROLLERS BY MEANS OF PETRI NETS, FSMS AND DATABASES

Agnieszka Węgrzyn, Arkadiusz Bukowiec

University of Zielona Góra, Institute of Computer Engineering and Electronics
ul.Podgórna 50, 65-246 Zielona Góra, Poland,
e-mail: {A.Wegrzyn, A.Bukowiec}@iie.uz.zgora.pl

ABSTRACT

In the paper the system for modeling and implementation of concurrent controllers is presented. Concurrent controllers are specified by Petri nets. Then Petri nets are decomposed using symbolic method of analysis. In the result the set of finite state machines (FSMs) is received. Each finite state machine is implemented using methods of structural decomposition during process of logic synthesis. There is applied method of multiple encoding of microinstruction in the approach presented in the paper. It leads to decreased number of Boolean function realized by combinational part of FSM and it leads to decrease of usage of logic elements of FPGA device. The additional decoder could be implemented using embedded memory blocks and it leads to balanced usage of hardware resources.

Index Terms **-** concurrent controllers, programmable logic controllers, Petri nets, digital systems, FSM, FPGA

1. INTRODUCTION

In the paper the system for modeling, analysis and implementation of concurrent controllers is presented. For modeling and analysis, Petri nets are used. For implementation process Finite State Machines (FSMs) are applied.

In the presented system, Petri net is used for specification and analysis of concurrent controllers. Such approach gives more possibilities, because there are a lot of method for formal verification of Petri net and Petri net in easy way show concurrency.

The concept of using Petri net for modeling digital circuits, especially concurrent controllers was proposed in 70s. Theory of Petri nets in join with method of formal logic give possibilities for verification of reprogrammable logic controllers systematic method [1].

During verification process of Petri net it is possible decomposition of the net into set of FSMs.

The Field Programmable Gate Arrays (FPGAs) are used very often to implement the logic circuits of FSMs [14]. The main feature of FPGA is an existence of look-up-tables (LUTs) with limited number of inputs [12]. On the other hand, logic functions of FSMs have much more arguments than typical LUT has inputs. This imbalance leads to need of decomposition [7, 12]. The negative results of functional decomposition are increasing a number of levels in the FSM circuit. Methods of structural decomposition [3, 6] permit to decrease a number of LUTs of FSM circuit in comparison with well-known design methods [2, 3, 7, 8, 12]. In the result there will be received two-level circuit. The decoder in second level of can be easy implemented with use of embedded memory blocks. It leads to balanced usage of hardware resources of FPGA device.

2. MODELLING OF CONCURRENT CONTROLLERS BY MEANS OF PETRI NET AND FSM

2.1. Basic notation of Petri net

Petri Nets model can be presented as an oriented bipartite graph with two subsets of nodes called places and transitions, where each node is linked with another by arc. Petri nets model can be described as:

$$\mathrm{PN} = (P, T, F, M_0)$$

where:

- P – is a set of places;
- T – is a set of transitions;
- M_0 – initial marking;
- $\mathrm{P} \cap T = \varnothing$;
- $\mathrm{F} \subseteq (P \times T) \cup (T \times P)$.

Set of input and set of output transitions into place p are defined, respectively as follow:

$$\bullet p = \{t \in T : (t, p) \in F\},$$

$$p\bullet = \{t \in T : (p, t) \in F\}.$$

Set of input and set of output places into transition t are defined, respectively as follow:

$$\bullet t = \{p \in P : (p, t) \in F\},$$

$$t\bullet = \{p \in P : (t, p) \in F\}.$$

A marking of a net is defined as a function M :

$$P \to \{0,1,2,...\}.$$

It can be considered as a number of tokens situated in the net places. Number of tokens in a place p for marking M is denoted as $M(p)$. A transition t is enabled and can fire if each input places have a token. Transition firing removes one token from each input place and adds one token to each output place of it.

Places of the net represents local state of the controllers, transitions present changes between states. All places marking in the same time, defining global state of controller. Tokens marking, that state of controller is active. Initial marking represent initial state of controller. Each fork transition is initial point of concurrent processes. Each join transition synchronize processes, which join in this point [11].

On Fig.1 example of interpreted Petri net specified concurrent controllers is presented.

There are eleven local states - places $(p1,...,p11)$ and eleven transitions $(t1,...,t11)$. By transition $t1$ are condition of firing $(/\ pr1 * x40)$ and output $y1$.

For verify modeled by Petri nets controllers, some properties of Petri net are checked. The main tasks of Petri net analysis are checking some properties of the net, i.e. liveness, boundedness, persistence etc. Boundedness and liveness are the properties which depend on occurrence deadlocks and traps in Petri nets and dependences between them [9, 13, 16].

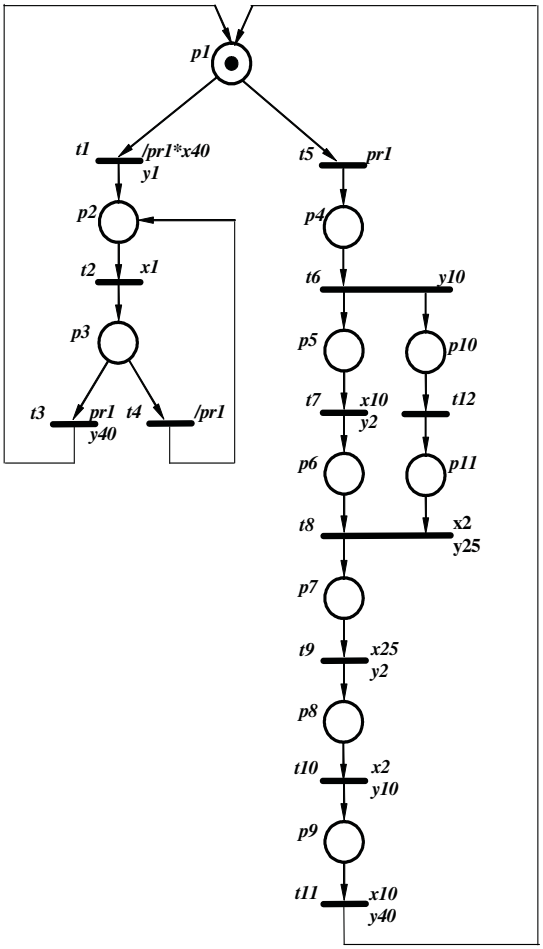

Fig. 1. Example of Petri net

2.2. Basic notation of FSM

Behavior of Mealy FSM can be described by a state diagram [7]. More formal representation of state diagram is direct structural table (DST) with columns [2, 8]: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h. Here a_m is an initial state of FSM, $a_m \in A$ where $A = \{a_1,\ldots,a_M\}$ is a set of internal states of FSM; $K(a_m)$ is a code of state $a_m \in A$ having $R = \lceil \log_2 M \rceil$ bits; a_S is a state of transition; $K(a_S)$ is a code of state $a_S \in A$; X_h is a conjunction of some elements from the set of logic conditions $X = \{x_1,\ldots,x_L\}$ that causes the transition $\langle a_m, a_s \rangle$; Y_h is a microinstruction formed under the transition $\langle a_m, a_s \rangle$, $Y_h \subseteq Y$, where $Y = \{y_1,\ldots,y_N\}$ is a set of microoperations; Φ_h is a set of excitation functions that are equal to 1 to switch the FSM memory from $K(a_m)$ to $K(a_s)$, $\Phi_h \subseteq \Phi$, where $\Phi = \{D_1,\ldots,D_R\}$ is a set of excitation functions; h is a number of transition ($h = 1,\ldots,H$). The internal variables $Q_r \in Q$, where $Q = \{Q_1,\ldots,Q_R\}$ is a set of internal variables, are used to encode internal states of FSM.

One of the most popular text format describing DST is the KISS2 format [17]. A file in this format consists of two parts: a header and a table (Fig. 2). The Header includes information about the number of inputs (L), the number of outputs (N), the number of table lines (H), the number of states (M) and the reset state. The last information is not required. The table describe behavior (transition) of FSM. It has four columns: a logic condition (X_h), a current state (a_m), a next state (a_s), output signals (Y_h).

This format does not give information about code of state and excitation functions. This information has to be generated during synthesis process and depend on particular synthesis method.

```
.i 2        # number of inputs
.o 3        # number of outputs
.p 12       # number of products
.s 6        # number of states
.r st0      # reset state
-0   st0    st0   000
-1   st0    st1   100
0-   st1    st0   000
1-   st1    st2   010
0-   st2    st1   010
11   st2    st3   001
10   st2    st3   101
-0   st3    st4   110
-1   st3    st3   -01
-0   st4    st5   110
-1   st4    st4   0--
--   st5    st0   ---
```

Fig. 2. An example of KISS2 file

3. THE IDEA OF THE DESIGN SYSTEM

Firstly concurrent controllers are modeled by Petri nets (in presented case using interpreted, hierarchical and colored Petri nets). Petri net is specified by textual format. In presented case PNSF3 format [15] or extended PNML format [10] is used. Then structure of Petri nets is transformed into relational database. Based on data from database, Petri net is decomposed into set of FSMs by algorithm described below. Decomposed FSMs is stored in database, too (Fig. 3).

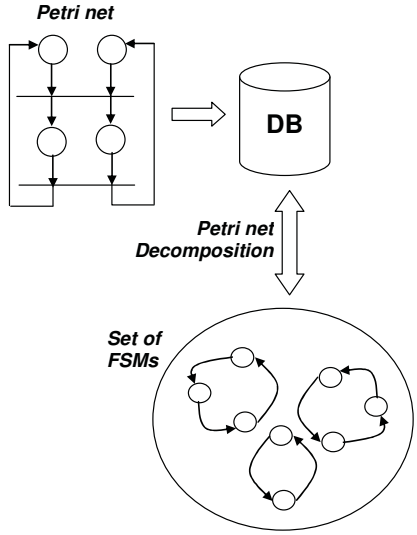

Fig. 3. Example of Petri net

3.1. From Petri nets to FSMs

In the section method of transformation Petri net into FSM is presented. The method based on decomposition of Petri net into automata using Thelen algorithm. In this section decomposition algorithm will described generally on an example (in detail was described in [9]). During decomposition a Petri net is divided into a set of subnets. These subnets have to satisfy some restriction, e.g. a subnet must include only places which are sequential to each other or cannot contain multi-input or multi-output transitions.

First step of presented method is generation Horn formulae for analyzed Petri net. Such formula is generated twice: once for deadlocks and once for traps.

Then for both equations Thelen trees are generated. Thelen has proposed an efficient algorithm for converting a conjunctive form into the sum of all prime implicates. It is based on building a search tree, such that every level of it corresponds to a clause of the CNF, and the outgoing arcs from a node correspond to the literals of the clause. Conjunction of all the literals corresponding to the arcs on the path from the root of the tree to a node is associated with the node. The tree is searched in DFS order, and several pruning rules are used to minimize it.

Basing on algorithm for finding all deadlocks and traps in Petri net and checking dependencies between sets of deadlocks and traps, it is possible to answer the question if Petri net is bounded and live.

Decomposition of Petri net can be based on coloring of Petri net. For decomposition and checking boundedness P-invariants are used. Subsets of places satisfying both Horn formulae for deadlocks and traps at the same time can be P-invariants. Each such vector is verified by incidence matrix. Each P-invariant is nonnegative integer vector satisfying the matrix equation $\gamma^T \bullet C = 0$, where C is the incidence matrix of the net.

Sets of deadlocks and traps are equal, i.e. each vector can be P-invariant. Results of multiplication of vector and incident matrix are 0, i.e. both vectors are P-invariants.

After calculation following P-invariants was received:

$[1,1,1,1,1,1,1,1,1,0,0]$

$[1,1,1,1,0,0,1,1,1,1,1]'$

which correspond to:

$(p1 \wedge p2 \wedge p3 \wedge p4 \wedge p5 \wedge p6 \wedge p7 \wedge p8 \wedge p9)$

$(p1 \wedge p2 \wedge p3 \wedge p4 \wedge p7 \wedge p8 \wedge p9 \wedge p10 \wedge p11)$.

Such P-invariant corresponds to one FSM and to one color in Petri net. The number of colors depends on number of P-invariants for flat Petri net. In hierarchical Petri nets it is not so clear. Based on such P-invariants Petri net was colored using two colors (Fig. 4).

If it is possible to color Petri net, i.e. that Petri net could be decomposed into FSM. P-invariants correspond to FSM subnets.

Each color corresponds to one FSM. In presented case color 1 represents FSM on Fig. 5a, color 2 represents FSM on Fig. 5b. Because places $p1, p2, p3, p4, p5, p6, p7, p8, p9$ are combined for such two FSMs, in one FSM occur these places, and the remainder FSMs are added additional place, which closes FSMs.

For firing transition t8, places p6 and p11 have to be marking. Because of this fact for transition t8 in first FSM (Fig. 5a) was added condition p11 and in the second FSM (Fig. 5b) condition p6.

3.2. Synthesis of FSMs

The Petri net was decomposed and two FSMs S_1 and S_2 were received (Fig. 6). The FSM S_1 (Fig. 6a) has 3 states: $SP, P10, P11$; 3 inputs: $x2, yp4, yp6$; 3 outputs: $y10, y25, yp11$. There is 5 transitions between states but the transition form the state $P11$ to the state $P11$ has expanded condition and it has to be represented as two lines (last two lines) in KISS2 file (Fig. 7a).

The FSM S_2 (Fig. 6b) has 9 states from $P1$ to $P9$; 7 inputs: $PR1, x1, x2, x10, x25, x40, yp11$; 7 outputs: $y1, y2, y10, y25, y40, yp4, yp6$. There is 18 transitions

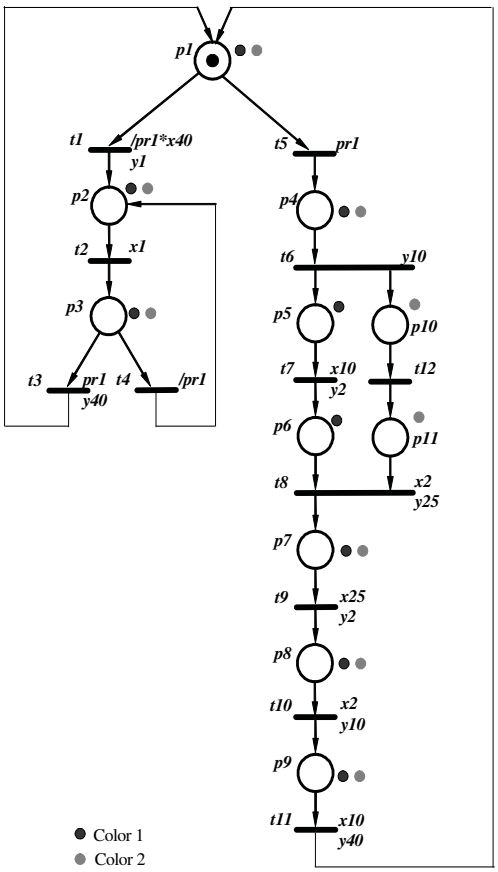

Fig. 4. Colored Petri net

between states but the transition $\langle P6, P6 \rangle$ has to berepresented as two lines in KISS2 file from the same reasons.

There was added additional signals $yp4, yp6, yp11$ in comparison with the original Petri net. The signal $yp11$ is generated in the FSM S_1 and it is an input of the FSM S_2. Signals $yp4, yp6$ are generated in the FSM S_2 and they are inputs of the FSM S_1. These three signals are used to synchronise work of both state machines.

Now the set FSMs can be synthesized into FPGA device. There can be considered several methods of synthesis [2, 3, 6, 12]. These methods are not described in this paper because they are widely discussed in literature and they are out of main scope of this paper.

Because the FSM S_1 has only 3 states and 3 inputs there is no need to apply method of structural decomposition because each function could be dependable maximum on 5 variables. There can be used trivial way of synthesis into single-level structure P with binary encoding of states [2, 3, 12]. The results of synthesis of the FSM S_1 for Xilinx Virtex V50 device are shown in Table 1.

The FSM S_1 has 9 states and it leads to 4 variables required for encoding states. There is also 7 inputs. It gives 11 variables and each output function and

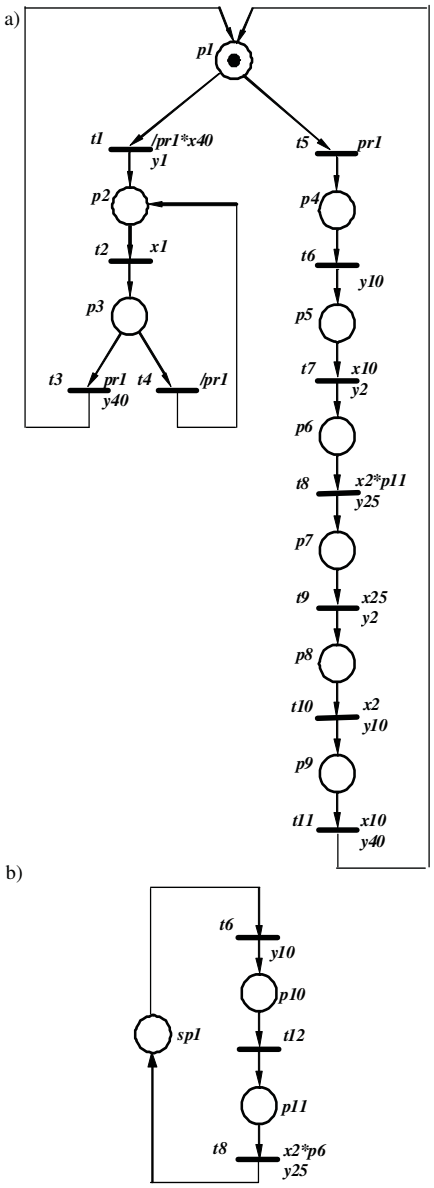

Fig. 5. Decomposed Petri net

excitation function can be dependable on these variables and there is 11 such functions in total. In this case each Boolean function has to be decomposed because one LUT cannot implement such big functions. But number of such function can be diminished by applying structural decomposition. In this case the method with multiple encoding of microinstruction into PY_0 structure was chosen [4]. The whole synthesis algorithm is not discussed in this paper and it can be found in literature. It can be also done automatically using The AS System [5]. The Table 2. shows results of synthesis of FSM S_2 for Xilinx Virtex V50 device. For comparison there was given also results for single-level structure P. The second level decoder of structures PY_0 was implemented using BlockRAM elements with initial content working in ROM mode. Because BlockRAM of Xilinx Virtex devices works as synchronous

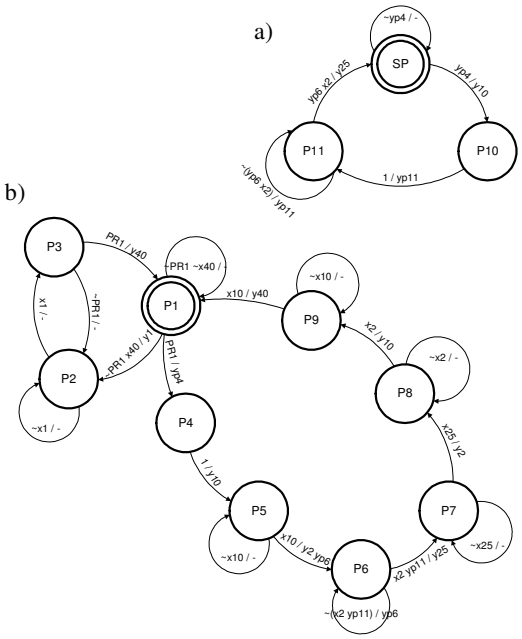

Fig. 6. Set of FSMs

memories there is no need to use additional registers for registered outputs.

To implement whole system there is required to create top-level module that connects set of FSMs together. In general, the top-level module should satisfy these conditions:

- All FSMs should have the same Clock and Reset signals;

a)
```
.i 3
.o 3
.s 3
.p 6
.r SP
-0- SP  SP  000
-1- SP  P10 100
--- P10 P11 001
1-1 P11 SP  010
0-- P11 P11 001
--0 P11 P11 001
```

b)
```
.i 7
.o 7
.s 9
.p 19
.r P1
0----0- P1 P1 0000000
0----1- P1 P2 1000000
1------ P1 P4 0000010
-0----- P2 P2 0000000
-1----- P2 P3 0000000
0------ P3 P2 0000000
1------ P3 P1 0000100
------- P4 P5 0010000
---1--- P5 P6 0100001
---0--- P5 P5 0000000
--1---1 P6 P7 0001000
--0---- P6 P6 0000001
------0 P6 P6 0000001
----1-- P7 P8 0100000
----0-- P7 P7 0000000
--1---- P8 P9 0010000
--0---- P8 P8 0000000
---1--- P9 P1 0000100
---0--- P9 P9 0000000
```

Fig. 7. KISS2 representation of the set of FSMs

Table 1. Synthesis results of FSM S_1

Logic Utilization	Structure	
	P	P (registered outputs)
Number of Slices	2	2
Number of Slice Flip Flops	2	3[1]
Number of 4 input LUTs	3	3
Number of bonded IOBs	8	8
Number of GCLKs	1	1

Table 2. Synthesis results of FSM S_2

Logic Utilization	Structure		
	P	P (registered outputs)	PY_0
Number of Slices	13	13	10
Number of Slice F-F	4	11	4
Number of LUTs	25	25	19
Number of IOBs	16	16	16
Number of BRAMs	0	0	1
Number of GCLKs	1	1	1

- All FSMs should have the same input signals (or subset of input signals);
- Additional output signals (for synchronization) should be connected with corresponding additional input signals in others FSMs;
- In case of an output signal could be generated under control from more than one component, corresponding output signals from all FSMs should be connected together in *final* output signal via OR-gate.

For our example the top-level module is presented in Figure 8.

Such system was synthesized into Virtex V50 device. The obtained results of device utilization are shown in Table 3 in the column FSMs P-PY_0. For comparison purpose in the column Petri Net structure are presented results after synthesis of behavioral description in Verilog HDL of initial Petri net.

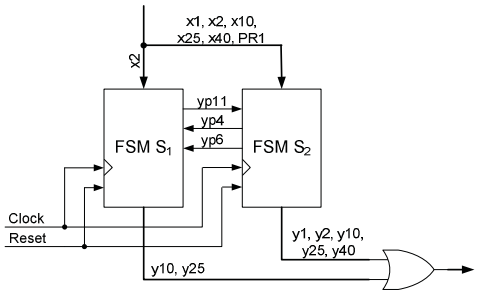

Fig. 8. The schematic of top-level module

[1] There is required only three flip-flops because two output functions are equal to excitation functions and they can use common registers.

Table 3. Synthesis results of whole system

Logic Utilization	Structure	
	Petri Net	FSMs P-PY$_0$
Number of Slices	15	12
Number of Slice Flip Flops	16	7
Number of 4 input LUTs	27	23
Number of bonded IOBs	13	13
Number of Block RAMs	0	1
Number of GCLKs	1	1

It is shown that decomposition of Petri net into FSMs and applying suitable method of synthesis reduce required hardware resources like LUTs and Flip-Flops.

4. CONCLUSIONS

The method of modelling and implementation of concurrent controllers proposed in this article simplify the whole process of its designing. Application of Petri nets allow easy modelling of concurrency. The conversion into sets of FSMs gives opportunity of applying advanced methods of logic synthesis. It leads decrease the number of logic elements required for implementation of the system in FPGA device and balanced usage of differed kind of logic elements.

The conversion from Petri net into FSMs can be obtained automatically based on the complex mathematical theory of Petri nets. Also the logic synthesis of FSMs is done automatically based on set and logic theory.

The application of data base allow to store all models in one place and invoke the third party tools for decomposition, synthesis and even remote programming of FPGA device.

5. REFERENCES

[1] M. Adamski, Projektowanie układów cyfrowych systematyczną metodą strukturalną, In Monografie, No. 49, WSI Press, Zielona Góra, 1990.

[2] S. Baranov, Logic Synthesis for Control Automata, Kluwer Academic Publishers, Boston, 1994.

[3] A. A. Barkalov, Synthesis of Operational Units, DonNTU, Donetsk, 2003.

[4] A. A. Barkalov, A. Bukowiec, "Synthesis of Control Unit with Multiple Encoding of the Sets of Microoperations", Proceedings of the International Workshop Discrete-Event System Design DESDes'04, ss. 75-78, 2004.

[5] A. Bukowiec, Automata Synthesis System, http://willow.iie.uz.zgora.pl/~abukowie/AS/as.htm, 2008.

[6] A. Bukowiec, Synthesis of Finite State Machines for FPGA Devices Based on Architectural Decomposition, In Lecture Notes in Control and Computer Science, Vol. 13, University of Zielona Góra Press, Zielona Góra, 2009.

[7] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw Hill, New York, 1994.

[8] D. Gajski, Principles of Digital Design, Prentice Hall, New Jersey, 1997.

[9] A. Karatkevich, Dynamic Analysis of Petri net-based Discrete Systems, In Lecture Notes in Control and Information Sciences, Vol. 356, Springer-Verlag, Berlin, 2007.

[10] E. Kindler, "PNML: Concept, Status and Future Directions", Entwurf Komplexer Automatisierung-ssysteme (EKA), Vol. 9, pp. 35-55, 2006.

[11] T. Kozłowski , E. L. Dagless, J. M. Saul, M. Adamski, J. Szajna, "Parallel controller synthesis using Petri nets", IEE Proceedings-E, Computers and Digital Techniques, Vol.142, No.4, pp. 263-271, 1995.

[12] T. Łuba, Synteza układów logicznych, WSISiZ, Warsaw, 2001.

[13] J. L. Peterson, Petri Net Theory and The Modeling of Systems, Prentice Hall, Englewood Cliffs, 1981.

[14] Z. Salcic, VHDL and FPLDs in Digital Systems Design, Prototyping and Customization, Kluwer Academic Publishers, Boston, 1998.

[15] A. Wegrzyn, Symboliczna analiza układów sterowania binarnego z wykorzystaniem wybranych metod analizy sieci Petriego, In Lecture Notes in Control and Computer Science, Vol. 3, University of Zielona Góra Press, Zielona Góra, 2003.

[16] A. Węgrzyn, "Parallel algorithm for computation of deadlocks and traps in Petri nets", Emering Technologies and Factory Automation - ETFA 2005, Vol. 1, pp. 143-148, 2005.

[17] S. Yang, Logic Synthesis and Optimization Benchmarks User Guide. Version 3.0, Technical Report, Microelectronics Center of North Carolina, Durham, 1991.