INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 4, PP. 397–402
Manuscript received November 11, 2012; revised Decemer, 2012. DOI: 10.2478/v10177-012-0054-y

Hierarchical Configurable Petri Net Modeling in VHDL

Michał Doligalski and Marian Adamski

***Abstract*—The paper presents method for hierarchical configurable Petri nets description in VHDL language. Dual model is an alternative way for behavioral description of the discrete control process. Dual model consists of two correlated models: UML state machine diagram and hierarchical configurable Petri net (HCfgPN). HCfgPN are Petri nets variant with direct support of exceptions handling mechanism. Logical synthesis of dual model is realized by the description of HCfgPN model by means of hardware description language. The paper presents places-oriented method for HCfgPN description in VHDL language.**

***Index Terms*—HCfgPN, UML state machine diagram, VHDL, logic controller.**

I. Introduction

PETRI nets, beside UML state machine [1], are one of the most popular model for logic controllers (LCs) behavioral description. Dynamic growth of discrete control systems, new functionality like partial reconfiguration justifies new approach to formal specification. Dual specification in form of UML state machine diagram and hierarchical configurable Petri net are one of the possible solutions. The application of two correlated models as one model simplify specification of logic controllers. Thanks to state machine diagram, such specification is transparent and understandable for all participants of the logic controller developing. The correlation of two models at specification stage brings out advantages of both component models eliminating, at the same time, their drawbacks [2]. Hierarchical configurable Petri nets are alternative solution for exceptions handling in Petri nets based specification [3] and enables direct mapping of state machine diagram. Thanks to Petri net formalism it is easy for formal verification and implementation using logic reasoning techniques [4] both for logical resources optimization and logic controller formal verification. Alternative approach is architectural decomposition of the Petri net into Finite State Machines [5] or synthesis of Petri nets by means of flexible memories [6]. The paper presents method for hierarchical configurable Petri nets behavioral description by means of VHDL language.

A. Logic Controller Developing Process

Classical approach for logic controller developing considers four stages: specification, synthesis, implementation, verification. This process is similar to well known waterfall approach to software developing. Unfortunately the application of waterfall process, both in software and LC developing, increase of risk of the project and move it to the end. Reduction risk and quality of final product, thought as LC, may be performed by several ways: formal verification, feedback to previously steps, post synthesis simulation or in circuit verification. In the proposed method, quality of logic controller is increased by developing a more precise model and (or) its formal verification. Device implementation of reconfigurable logic controllers (RLCs) with the use of Field Programmable Gate Array devices (FPGA) is a quite commonly used solution. Research covering both specification based on behavioral models [7], formal verification [8], [9] as well as the controller synthesis [10]–[12] are carried out within a broad range.

The research was financed from budget resources intended for science in 2010 – 2013 as an own research project No. N N516 513939.

M. Doligalski and M. Adamski are with the Computer Engineering & Electronics Department, University of Zielona Góra, ul. Licealna 9, 65-417 Zielona Góra, Poland (e-mails: {m.doligalski; m.adamski}@iie.uz.zgora.pl).

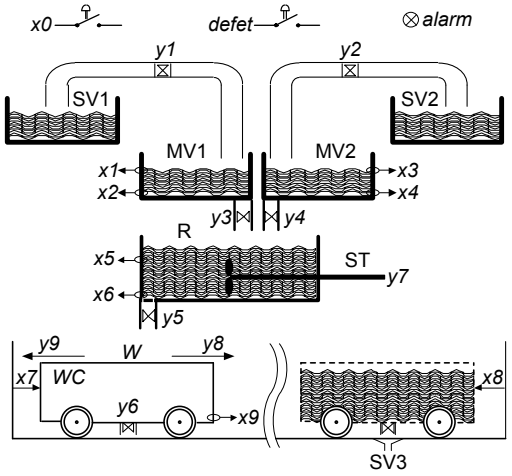

Fig. 1. Outline of production process [13].

II. Example of Control Process Specification

The aim of the production process (Fig. 1) is mixing liquid substances and transporting final product using cistern. Production process is controlled by logic controller (LC), with eleven inputs ($x_1..x_9, defect$) and ten outputs ($y_1..y_9, alarm$). LC inputs are connected with sensors and switches ($x_0, defect$), outputs with actuators and defect indicator (lamp – $alarm$). Starting material is kept in vats: $SV1$ and $SV3$. Measuring of material volume is performed in vats: $MV1$ and $MV2$. Final product is mixed in reactor tank R and poured into tank M.

System behavior was formally described by means of UML state machine diagrams (Fig. 2). Three levels of hierarchy and two parallel states were identified. Orthogonal state *Process*

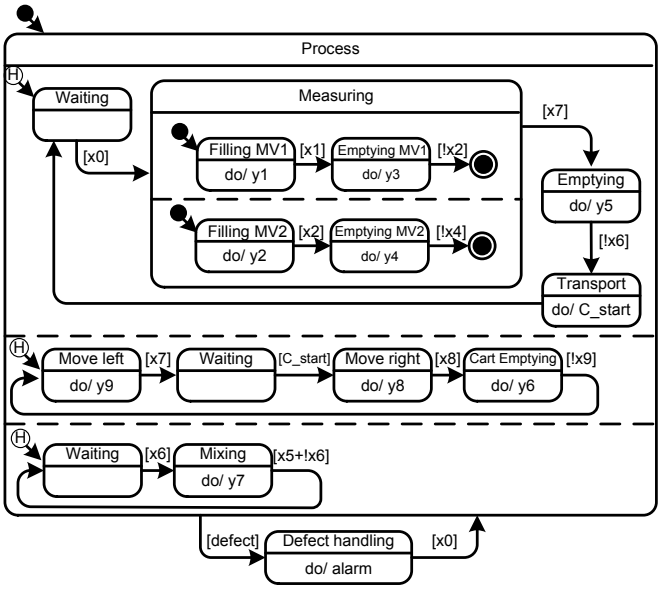

Fig. 2. State machine diagram.

implements shallow history. It means that if preemption occurs (exception) last marking will be preserved, and next, after resumption, control process will continue from place where was stopped. In this particular case, exception is triggered by signal *defect*. Resumption of the control by signal x_0.

Specification includes internal signal *C_start*, responsible for the communication between two concurrent regions. Additional signal is not necessary, but allows and simplifies cistern movement during measuring and mixing new portion of materials. Current UML language standard does not define synchronization, but this functionality may be implemented by means of signal broadcasting.

State machine diagram acts as front-end, user do not have to worry about Petri net formalism. The transformation of state machine diagram to Petri net diagram is done automatically, with the use meta-models transformation methods. Presented state machine diagram consists of three sub-machines. During transformation, for each submachine one subnet will be created.

Diagrams (Figs. 3, 4, 5) presents main net (top level net) and two subnets, each for one macro-place. Diagrams have been prepared in accordance with the assumptions of HCfgPN formalism. In comparison to interpreted Petri nets, functionality of transition triggering, token movement and output signal generation was redefined.

Each subnet consists of two parts (blocks, subnet): operational and configurational. The operational subnet (OS) is responsible for control algorithm implementation. Operational subnet is controlled by configurational subnet (CS), which is responsible for exceptions handling (preemption) and resumption. The CS consists of three configurational places (P^{init}, P^a, P^i) and five transitions ($T^{init}, T^a, T^i, T^w, T^{fin}$). When place P^a is marked, token movement and output signal generation in OS is possible, unlike when the place P^i is marked. Transitions T^w firing removes (kills) all tokens from OS and move subnet to initial marking.

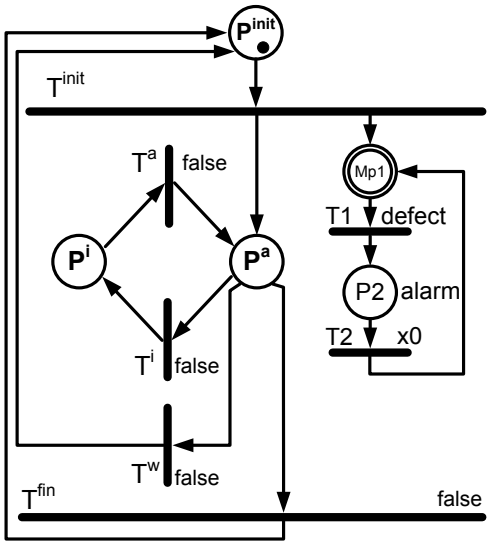

Fig. 3. Top level net.

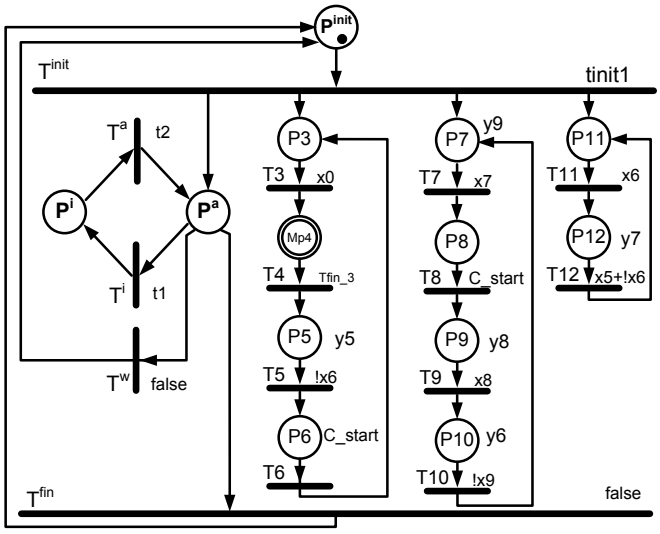

Fig. 4. Subnet 2 – extension of macroplace MP1.

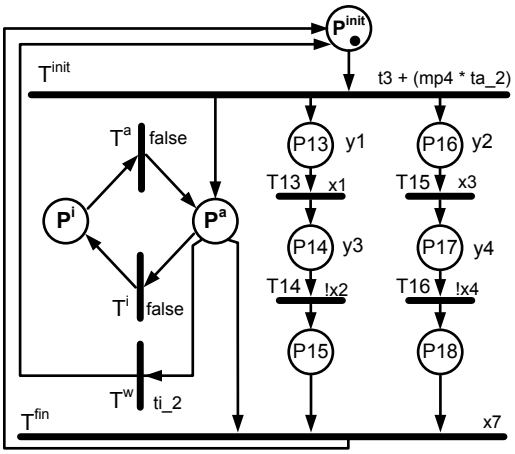

Fig. 5. Subnet 3 – extension of macroplace MP2.

```
transitions:process (P3, MP4, P5, ... , P12,
    x0, ..., x9, t1, t2, tfin_3, tinit_1,
    C_start, tfin_3, LocalConfig)
Begin
if P3='1' and x0='1' and LocalConfig='1' then
    T3<='1'; else T3<='0';end if;
f MP4='1' and x7='1' and tfin_3='1'  and
    LocalConfig='1' then T4<='1'; else T4
    <='0';end if;
if P5='1' and x6='0' and LocalConfig='1' then
    T5<='1'; else T5<='0';end if;
...
 end process;
```

Fig. 6. Template of operational transition description in VHDL.

Several transitions in Figs. 3 – 5 has a trigger specified as *false*. Those transitions do not participate in the process and are not necessary, but intentionally were left in the diagram to show the structure of CS and OS subnets. This transitions and superfluous places connected with them will be removed during optimization process at synthesis and implementation phase.

Automatic (non guarded and not triggered) transition in state machine diagram, correspond to final transition firing in Petri net model. Transition T^fin from subnet 3 (Fig. 5) finalize macroplace realization. Transition $T4$ from subnet 2 (Fig. 4) is waiting for T^fin enabling transition and trigger at same time.

III. Description in VHDL Language

Petri net behavioral functional modeling by means of Hardware Description Languages (HDLs) is a well grounded subject and a number of solutions have been presented [14]–[19]. For VHDL and Verilog languages two main methods of description were developed: place-oriented and transition-oriented. There is also third method of description that mix previous technics transitions and places oriented. Also the structural techniques have their place in the description of Petri nets [20].

In the transitions-oriented technics, Petri nest is described as set of transition with guards and input arcs. In each step, synchronized by clock, fulfilment of firing guards results transitions firing and new net marking. Petri net described is as one process that is responsible for transition firing possibility checking and next marking generation.

In the place-oriented technics, two processes implements control algorithm functionality. One process is responsible for checking of transitions firing possibility, second is responsible for generation of the next marking for the net. This method of description, due to separation into two process, seems to be more transparent. Also chip and very large FPGA devices, in the connection with very good tools for synthesis and implementation, makes failing to investigate the efficiency of resource use for particular method justified. The paper will present VHDL approach for HCfgPN description, place-oriented technics in particular.

General structure of the HCfgPN (Net_2) description in VHDL language was presented in Fig. 8. In the proposed

```
opeational_places:process (CLK, RESET)
Begin
if RESET='1'  then
        P3  <='0';
        MP4 <='0';
        P5  <='0';
...
 elsif CLK'event and CLK='1'then
     if Tw ='1'  then
          P3  <='0';
          MP4 <='0';
          P5  <='0';
...
     else
     if Ti='0' and Pi='0'   then
     if Tinit='1' or T6='1' or (P3='1' and T3
         ='0') then P3<='1'; else P3<='0';end
         if;
     if T3='1' or (MP4='1' and T4='0') then MP4
         <='1'; else MP4<='0';end if;
     if T4='1' or (P5='1' and T5='0') then P5
         <='1'; else P5<='0';end if;
...
     end if;
     end if;
end if;
end process;
```

Fig. 7. Template of operational places description in VHDL.

solution, based on classical method of description [17], [19], each subnet is described by four processes. Two processes are responsible for configurational transitions and places, next two processes for operational subset mapping. Signal *Local config* informs operational processes if it is possible to move token between places.

Process *config Transitions* describes all operational transitions and is responsible for initial marking initialization in case of: preemption (T^w firing), final transition firing (t^{fin}). The resumption is realized by T_a transition firing. In case of Net_2 subnet, shallow history option is desired – t^w transition is switched off by adding of *false* ("0") value to transition guard.

Critical and noncritical exceptions in case of UML state machine diagram are modeled by means of history pseudostate. Proposed structure of HCfgPN description give the possibility of preemption description also with history (resumption). Place P^i, in case of preemption with history (noncritical exception), catch token from place P^a and hold it till resumption condition will occur (transition T^a fires).

In case of critical exception, guard should be assigned to transition T^w, all tokens from places located in OS and move subnet into initial marking.

Operational transition are described as typical process section (Fig. 6). Sensitivity list of this process consists of set predicates and places that are involved in operational transition enabling. Each transitions is formed as functional description with the use *If* clause, when condition is formed as logical conjunction of input places, guard condition and *LocalConfig* signal. This signal blocks transition firing when subnet is inactive (place P^a is not marked) or when transitions with higher priority will fire (transitions :T^i, T^w).

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
Entity Net_2 is
port(CLK, RESET, x0, x5, x6, x7, x8, x9, t1,
    t2, tfin_3, tinit_1 :in  std_logic;
          Ti_out, Ta_out, T3_out, MP4_out, y5,
              y6, y7, y8, y9 : out std_logic) ;
End Net_2_VHD;
Architecture Net_2 of Net_2 is
Signal P3, MP4, P5, ... , P12 : std_logic
    :='0';
Signal T3, ... ,T12: std_logic:='0';
Signal Pinit: std_logic:='1';
Signal Pa, Pi, Tinit, Tfin, Ti, Ta, Tw,
    LocalConfig : std_logic:='0';
Signal C_start :std_logic:='0';
begin
Ti_out <=Ti; --local signals
...
y5              <= Pa and P5 ;--outputs
C_start <= Pa and P6;
...
LocalConfig <= Pa and  not (Ti or Tw);
config_Transitions:process(Pi, Pa, Ti, Tw,
    Pinit, t1,t2, P7, P9, t5, t6, tinit_1)
     begin
     Tinit <= Pinit and tinit_1 ;
     Tfin  <=  Pa and P7 and P9 and not(Ti and
         Tw);
     Ti <= Pa and t1 and  not Tw;
     Tw <= Pa and  '0';
     Ta <= Pi and t2;
          end process;

config_places:process (CLK, Reset)
Begin
if RESET='1' then
     Pinit<='1';
     Pi   <='0';
     Pa   <='0';
elsif CLK'event and CLK='1'then
     if Tw='1' or Tfin='1' or (Pinit='1' and
         Tinit='0') then Pinit<='1'; else Pinit
         <='0';end if;
     if Tinit='1' or Ta='1' or (Pa='1' and (Ti
         ='0' and Tw='0' and Tfin='0')) then Pa
         <='1'; else Pa<='0';end if;
     if Ti='1' or (Pi='1' and (Pi='0' or Ta
         ='0')) then Pi<='1'; else Pi<='0';end
         if;
end if;
end process;
operational_transitions:process (P3, MP4, P5,
    ... , P12, x0, ... , x9, t1, t2, tfin_3,
    tinit_1, C_start, tfin_3, LocalConfig)
Begin
...
end process;
operational_places:process (CLK, RESET)
Begin
  ...
end process;
end architecture;
```

Fig. 8. Template HCfgPN description in VHDL.

The higher priority of configurational transitions is necessary for proper exceptions handling: when undesirable occurs, control process should be held or stopped immediately.

Operational places process (Fig. 7) describes token movement between places in synchronous way. In the example, asynchronous reset for operational and configurational places was presented but it's also possible to describe it as synchronous event. An important aspect is the deterministic behavior of the network, so for the entire project should be selected one mode reset and applied consistently for each subnet. Transition T^W removes (kills) all tokens from operational subnet. Each operational place was described as logic conjunction of activation condition and maintain condition. In comparison to traditional technics, except of synchronous preemption reset (T^W), there is no difference in places description.

Outputs functions are logic conjunction of place P^a and alternation of places, where signal should be generated. This expression guarantees outputs signal generation only in case when subnet is active. Preemption removes token, both from operational places and P^a place and stops signal generation.

A. Macroplaces and Subnets

The UML state machine diagram is based on modularity. It can contain composite states, that are composed of one (composite state) or many (orthogonal state) submachines. This structure has to be reflected in the Petri net (HCfgPN). Composite state has its equivalent in the form of macroplace. Submachines that are contained in composite state are transformed as (paralel) path in operational part of the subnet.

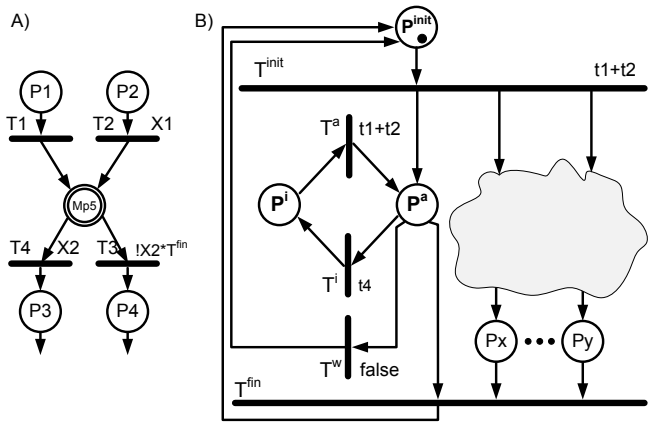

Fig. 9. Example of subnets communication.

Communication between subnets is implemented by signal exchange. Fig. 9 presents example of two subnets and connection between them. Subnet A contain macroplace $MP5$ with two inputs and two outputs arcs. Macroplace is connected with subnet B. Considered example describes noncritical exception, as it was presented in Fig. 2.

Initial transition (T^{init}) from subnet B contains condition formed as logical alternation of input transitions ($t1$ and $t2$). The same condition was assigned to transition T^a – resumption after noncritical exception should result in token movement from place P^i to P^a. Transition T^i firing will result with operational subnet (cloud in Fig. 9) deactivation (freeze).

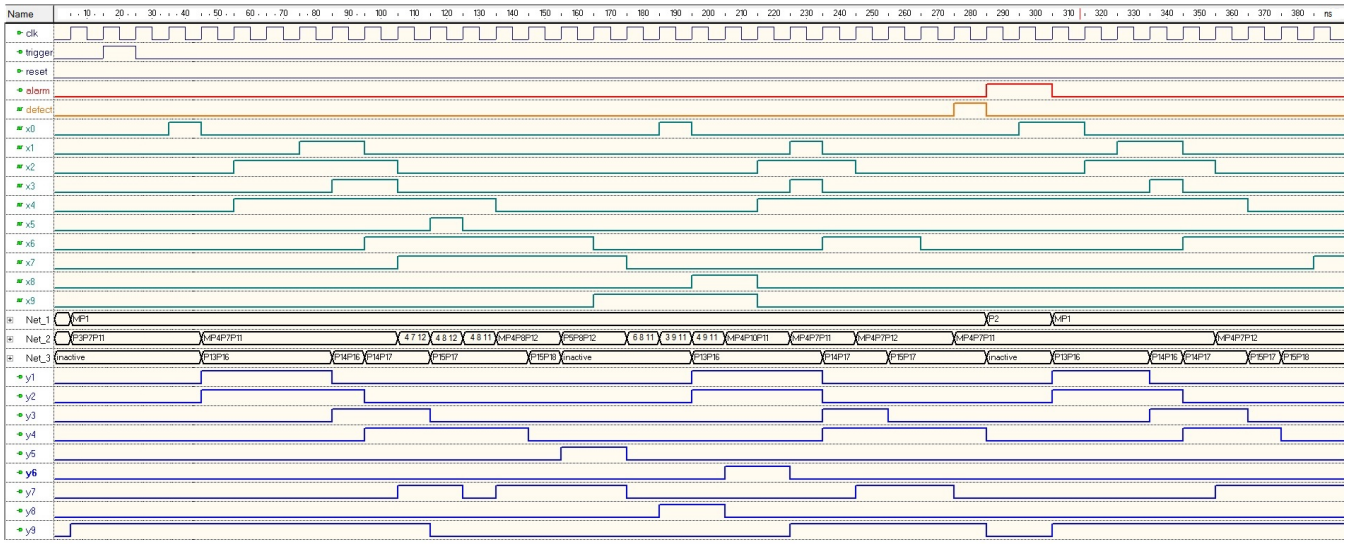

Fig. 10. Simulation of the logic controller in Active-HDL tool.

As the noncritical exception is considered, Transition T^i condition will be formed as alternation of outgoing transition with trigger from $MP5$. In case of critical exception, condition should be assigned to transition T^W.

Untriggered transition from macroplace $MP5$ will fire when all synchronization states from subnet B will be marked. For proper untriggered transition handling, T^{fin} transition from subnet should be placed as condition of outgoing transition on the supernet. From one place or macroplace, there can be only one outgoing untriggered transition.

IV. Implementation and Simulations Results

Proposed method requires separate RTL block for each subnet. Communication via signal broadcasting between block is allowed. RTL level schematic for considered logic controller was presented in Fig. 11. For each subnet one RTL block, understood as VHDL entity, was prepared. Synthesis and implementation was carried out using the Xilinx ISE software, results were presented in Tab. I.

TABLE I
Synthesis Result for Xilinx XC2VP30

Logic utilization	Available	Used (VHDL)
Number of Slices	13696	33
Number of Flip Flops	27392	25
Number of 4 input LUTs	27392	63
Number of bonded IOBs	556	23
Number of GCLKs	16	1

The previous attempts to mapping resume mechanism for hierarchical concurrent finite state machine (HCFSM) causing great waste of hardware resources. In state mirroring method every state has a copy of which stores information when the state machine to which it is expropriated [21], [22]. Presented approach [23], limits additional places in each size subnet to three (P^{init}, P^a, P^i). Of course, reducing the number of used flip-flops is paid with the increasing use of LUT tables. One hot encoding was used. As it was expected, = places P^i

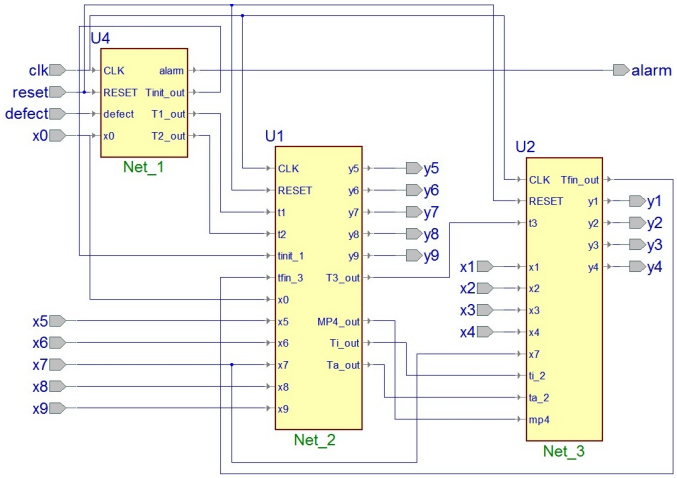

Fig. 11. RTL schematic of discrete logic controller.

from $Net1$ and $Net2$ were trimmed by synthesis tool during optimization process.

Simulation results were presented in Fig. 10. An 275 nanosecond noncritical exception occurs. Subnet 3 was pre-empted and moves to initial marking. Signal generation form subnet 2 was freezed, but last positions of tokens were remained. After resumption triggered by signal $X0$ subnet 2 was reactivated and output signal were enabled. As it can be seen in Fig. 10. Detailed simulation confirm the assumed implementation of the controller functionality.

V. Conclusion

Dual specification is a alternative method for discrete control system description. Dual specification assumes the use of two correlated models on the stage of the specification: UML state machine and hierarchical configurable Petri net. The application of methods for transformation beetwen models at metamodel level allows for the transformation of models automatically, without user intervention.

The possibility of practical application is conditioned by by the availability of methods for the behavioral synthesis of the model a the level of the hardware description languages. Proposed method supports aspects as concurrent processes, exceptions handling and hierarchy modeling. In this paper places – oriented approach to the HCfgPN modeling in VHDL language was presented.

Variety of formal methods for Petri net verification empowers the application in rigorous methodology for safety critical systems developing [24].

Logic controller designed according to the presented methodology can be used either as a standalone module used for controlling the production process as well as part of embedded systems: System on Programable Chip (SOPC).

Future work should include a determination of synthesis methods using other forms of description, and in particular the structural synthesis in FPGA. Also comparison of effectiveness of modeling methods, synthesis and implementation tools will be an important aspect.

At present, optimization of hardware resources is not so critical aspect. It should, however, consider the question of energy savings, eg by temporarily disconnecting the individual subnets. Proposed structure of HCfgPN gives the potential for energy efficiency by disabling and enabling particular subnet.

REFERENCES

[1] G. Łabiak, M. Adamski, M. Doligalski, J. Tkacz, and A. Bukowiec, "UML modelling in rigorous design methodology for discrete controllers," *International Journal of Electronics and Telecommunications*, vol. Vol. 58, no. no 1, pp. 27–34, 2012.

[2] M. Doligalski, *Behavioral specification diversification of reconfigurable logic controllers*, ser. Lecture Notes in Control and Computer Science. Zielona Góra: University of Zielona Góra Press, 2012, vol. 20.

[3] ——, "Behavioral specification of the logic controllers by means of the hierarchical configurable petri nets," in *Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems - PDeS 2012*. Brno, Czechy: Brno, 2012, pp. 80–83.

[4] M. Adamski and J. Tkacz, "Formal reasoning in logic design of reconfigurable controllers," in *Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems PDeS'12*, Brno, Czech Republic, 2012.

[5] A. Bukowiec, "Synthesis of FSMs based on architectural decomposition with joined multiple encoding," *International Journal of Electronics and Telecommunications*, vol. 58, no. 1, pp. 35–41, 2012, DOI: 10.2478/v10177-012-0005-7.

[6] A. Bukowiec and M. Adamski, "Synthesis of Petri nets into FPGA with operation flexible memories," in *Proceedings of the IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems DDECS'12*, Tallinn, Estonia, 2012, pp. 16–21.

[7] A. Salihbegovic, Z. Cico, V. Marinkovi, and E. Karavdi, "Software engineering approach in the design and development of the industrial automation systems," in *the 2008 International Workshop on Software Engineering in East and South Europe*, ser. SEESE '08. New York, NY, USA: ACM, 2008, pp. 15–22.

[8] T. Kropf, *Intro. to Formal Hardware Verification*. Berlin: Springer-Verlag, 1999.

[9] J. Tkacz, "Deadlocks detection in logic controllers by means of using gentzen reasoning method (in Polish)," *Measurement Automation and Monitoring*, vol. 6, no. 6, pp. 11–13, 2006.

[10] E. Hrynkiewicz, A. Milik, and J. Mocha, "Dynamically reconfigurable concurrent implementation of the binary control (in Polish)," *Electronics: Constructions, Technologies, Applications*, vol. 49,(11), pp. 187–190, 2008.

[11] M. Adamski, "Specification and synthesis of petri net based reprogrammable logic controller," in *Programmable Devices and Systems 2001 (PDS 2001) : a Proceedings Volume From The 5th IFAC Workshop*, W. Ciazynski, E. Hrynkiewicz, and P. Klosowski, Eds. London: Pergamon, 2001, pp. 95–100.

[12] G. Łabiak and G. Borowik, "Statechart-based controllers synthesis in FPGA structures with embedded array blocks," *International Journal of Electronics and Telecommunications*, vol. 56, no. 1, pp. 13–24, 2010, DOI: 10.2478/v10177-010-0002-7.

[13] M. Adamski and J. Tkacz, "Formal reasoning in logic design of reconfigurable controllers," in *Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems - PDeS 2012*, Brno, 2012.

[14] J. Fernandes, M. Adamski, and A. Proena, "Vhdl generation from hierarchical petri net specifications of parallel controllers," *IEE Proceedings - Computers and Digital Techniques*, vol. 144, no. no 2, pp. 127–137, 1997.

[15] M. Adamski, M. Węgrzyn, and P. Wolański, "A vhdl based approach to logic controllers design," in *International Conference Programmable Devices and Systems - PDS'98 : Conference proceedings*, Institute of Elektronics, Silesian Technical University - Gliwice, Departament of Measurement and Control - Technical University Ostrava, Polish Section IEEE. Gliwice, Polska: Gliwice, Silesian University of Technology, 1998, pp. 9–16.

[16] M. Adamski and M. Węgrzyn, "Petri nets mapping into reconfigurable logic controllers," *Electronics and Telecommunications Quarterly*, vol. 55, no. 2, pp. 157–182, 2009.

[17] P. Wolański, "Modeling digital circuits at the rtl level using a subset of petri nets and vhdl (in Polish)," Ph.D. dissertation, Warsaw University of Technology, Faculty of Electronics and Information Technology, 1998, supervisor: Prof. dr hab. inż. Marian Adamski.

[18] M. Węgrzyn, "Hierarchical implementation of digital concurrent controllers using FPGA (in Polish)," Ph.D. dissertation, Warsaw University of Technology, Faculty of Electronics and Information Technology, 1998, supervisor: dr hab. in. Marian Adamski.

[19] M. Puczyńska, G. Łabiak, and P. Wolański, "Program implementation of the conversion of petri nets into vhdl (in Polish)," in *Reprogramowalne Układy Cyfrowe - RUC 2000 : Materiały III Krajowej Konferencji Naukowej*. Szczecin, Polska: Szczecin, Publishing and Printing Technical University Faculty of Computer Science, 2000, pp. 285–291.

[20] M. Bolton, *Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej*, 1991, ch. VHDL and its Use in VLSI Design, pp. 149–158.

[21] G. Bazydło and M. Adamski, "Exception handling in UML state machine implemented in digital microsystems (in Polish)," *Measurement Automation and Monitoring*, vol. 56, no. 7, pp. 728–731, 2010.

[22] M. Adamski and G. Bazydło, "Modeling emergency situations in a hierarchical state machine UML 2.4 with the use of the attribute history (in Polish)," in *Design, Analysis and Implementation of Real-Time Systems*, L. Trybus and S. Samolej, Eds., Warszawa, 2011.

[23] M. Doligalski and M. Adamski, "Exception handling mechanism and controll resumption in hierarchical petri nets (in Polish)," *Measurement Automation and Monitoring*, vol. 57, no. 6, pp. 671–674, 2011.

[24] A. Karatkevich, *Dynamic Analysis of Petri Net-based Discrete systems*, ser. Lecture Notes in Control and Information Sciences. Berlin: Springer-Verlag, 2007, vol. 356.