From UML Specification into FPGA Implementation

Grzegorz BAZYDLO[1], *Marian ADAMSKI*[1], *Marek WEGRZYN*[1], *Alfredo ROSADO MUNOZ*[2]

[1]Institute of Computer Engineering and Electronics, Faculty of Electrical Engineering, Computer Science and Telecommunications, University of Zielona Gora, Szafrana 2, 65-516 Zielona Gora, Poland
[2]Department of Electronic Engineering, School of Engineering (ETSE), University of Valencia, Avinguda de la Universitat s/n, 46100 Burjassot, Valencia, Spain

g.bazydlo@iie.uz.zgora.pl, m.adamski@iie.uz.zgora.pl, m.wegrzyn@iie.uz.zgora.pl, alfredo.rosado@uv.es

Abstract. *In the paper a method of using the Unified Modeling Language for specification of digital systems, especially logic controllers, is presented. The proposed method is based mainly on the UML state machine diagrams and uses Hierarchical Concurrent Finite State Machines (HCFSMs) as a temporary model. The paper shows a way to transform the UML diagrams, expressed in XML language, to the form that is acceptable by reconfigurable FPGAs (Field Programmable Gate Arrays). The UML specification is used to generate an effective program in Hardware Description Languages (HDLs), especially Verilog.*

Keywords

FPGA, FSM, logic controllers, UML, Verilog.

1. Introduction

There are many diverse methods of digital system design (e.g. [1]). Nowadays, the dominant trend is to design integrated circuits for a particular purpose (e.g. [2]). Millions of logic gates combined with the ability of reprogramming and reconfiguring [3] give unprecedented computational power, and the designed systems are becoming increasingly complex. Therefore, effective system design is supported by Computer Aided Design (CAD) systems and abstract modeling seems to be ever more important. Even higher consumption of basic electronic components seems to be acceptable because the chip costs are on the decrease.

The main purpose of this paper is to present a method of using UML state machines diagrams for graphic specification of digital systems, especially logic controllers. The reconfigurable FPGA devices can be used as a target implementation platform.

2. Proposed Design Method

The proposed design method attempts to combine advantages of graphical notation (readability, convenience, intuitiveness) with the benefits of hardware description language (precision, versatility). Figure 1 presents the flowchart of the proposed method. As a background, tools used by designer are presented. In the first step, the designer, using a UML editor, creates a behavioral description of the system with UML state machine diagram. Next, the diagram is exported to XML (the export option is built in every professional UML editor, e.g. IBM Rational Software Architect, Sparx Systems' Enterprise Architect). XML files, compliant with the XML Metadata Interchange (XMI) specification, are the input to the developed U2V system [4]. The system translates the UML diagrams to synthesizable HDL description, using temporary HCFSM model (each module is saved in a separate file). Then, using external tools, the system can

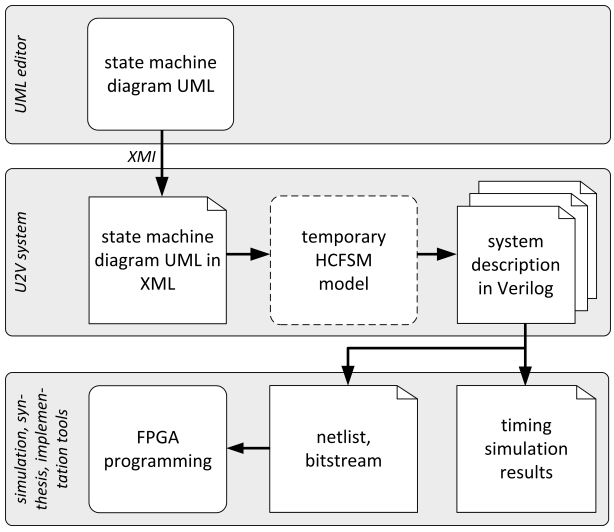

Fig. 1: Flowchart for the proposed design method.

be simulated, and finally the synthesis and implementation can be done. The result of the last step is the netlist and the related bitstream, whereby it is possible to program the real FPGA device.

3. Logic Controller Example

Figure 2 presents a mixer machine for beverage production and distribution (Mixer). The example is taken from [5]. The controller works in the following way: pressing the start button ($x1$) initiates the processes in which tanks 1 and 2 are being filled, and the containers for the beverages are delivered (signal $y3$). Active signal $x4$ means that containers have been placed correctly on the trolley. Then valves $y10$ and $y11$ are opened until the tanks are filled, and this information is indicated respectively by sensors $x5$ and $x7$. In turn, the delivery of the containers is connected with the movement of the trolley with containers (active signal $y12$) and finishes when the trolley reaches sensor $x13$. After filling the tanks, the ingredients are being prepared, which is initiated with signals $y1$ and $y2$. An indication of the sensors: $x2$ for the first container and $x3$ for the second container means that the ingredients in tanks 1 and 2 were prepared. The ready components are poured into the third tank by opening valves $y5$ and $y6$ and mixed (active signal $y4$ until deactivation $x9$). The valves are closed after emptying the tanks 1 and 2. The situation is signaled with sensors $x6$, $x8$ and $x9$ respectively. When one of the containers is ready, it is independently filled and closed (signals $x10$ and $x11$). After the completion of both processes, the trolley with containers moves back to its initial position, which is signaled with $y9$. Sensor $x12$ is active when the system is ready for the further operation.

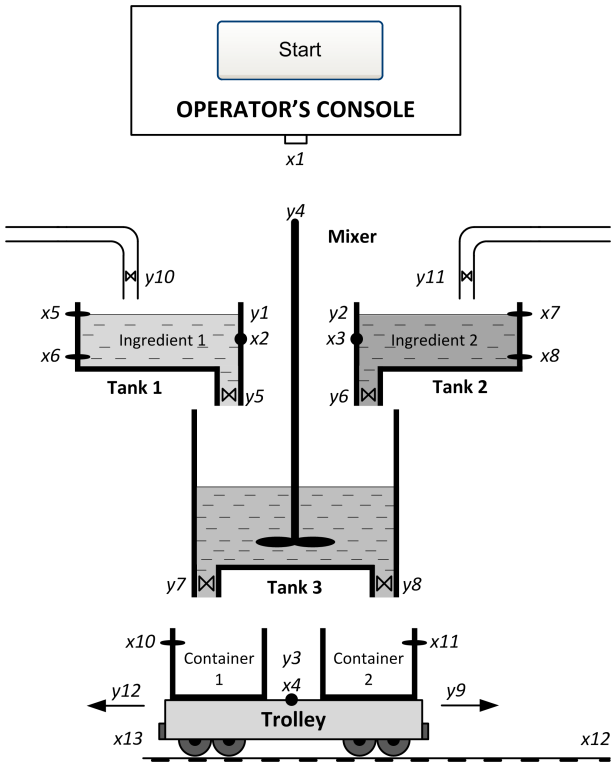

Fig. 2: Industrial mixer process diagram.

Formally, the presented controller consists of thirteen input sensors and twelve output signals. Controller's block diagram is shown in Fig. 3.

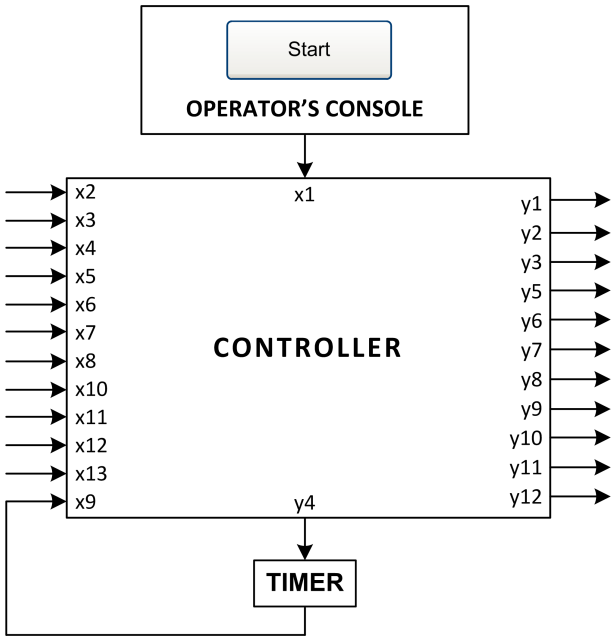

Fig. 3: The controller's block diagram for industrial mixer.

4. Unified Modeling Language

The Unified Modeling Language (UML) [6] contains a few graphical tools that can be used to illustrate, specify, visualize, construct and document artifacts of software and non-software systems [7], especially digital devices [8]. The first version of the language (1.0) was presented in 1995, and the current version is 2.4.2 [8]. The UML contains fourteen kinds of diagrams (structural and behavioral). One of them is a state machine diagram that defines a set of concepts that can be used for modeling behavior of the discrete systems, because they refer directly to the definition of Finite State Machines (FSMs). Graphically the state machine is represented as a directed graph with nodes related to states and edges related to transitions between states. In other words, it is a graphical representation of a discrete behavior of state-transition systems [9].

5. Using State Machine Diagrams

State machines can be used to express the behavior of a part of a system. The formalism of state machines described in UML is an object-based variant of Harel statecharts [10]. On the diagram, a state models a situation during that some invariant condition holds. The invariant may represent a static situation such as an object waiting for some external event to occur. However, it can also model dynamic conditions. A transition is a directed relationship between source (state) and target vertices. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to the occurrence of an event of a particular type.

State machine diagrams enable the designer to model the system on the selected hierarchical level. If there is no need to present all details of the designed system, it is able to choose a higher level of hierarchy and hide unimportant (on the designed level) information. Figure 4 shows a state machine diagram for Mixer's model on the highest hierarchy level. In the example, states "Beverage preparation and the movement to the left" and "Filling of containers" are, in fact, composite states. Composite states may consist of sequential or orthogonal substates.

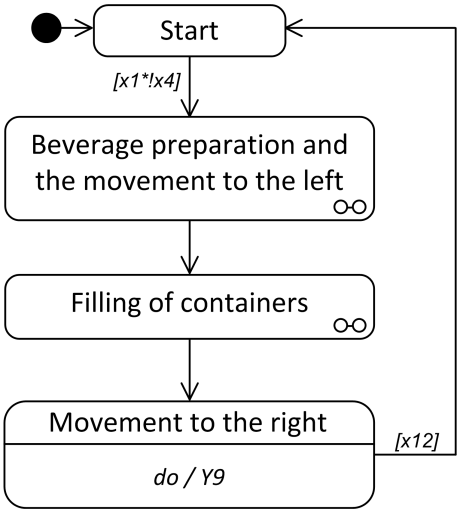

Fig. 4: A state machine diagram - the highest hierarchy level.

Figure 5 shows the state machine diagram on the lowest hierarchy level with all substates and full information. Each substate (that, in fact, is a sequential automaton) has its own initial state (on the diagrams: small, black circle) that is activated when the system is in a superior state.

It seems that the state machine diagrams are the most effective and important UML diagram for the graphic specification of digital systems. With the support of concurrency and hierarchy of the design system and the precise semantics, state machine diagram allows one to develop a complete and unambiguous description of the digital system behavior.

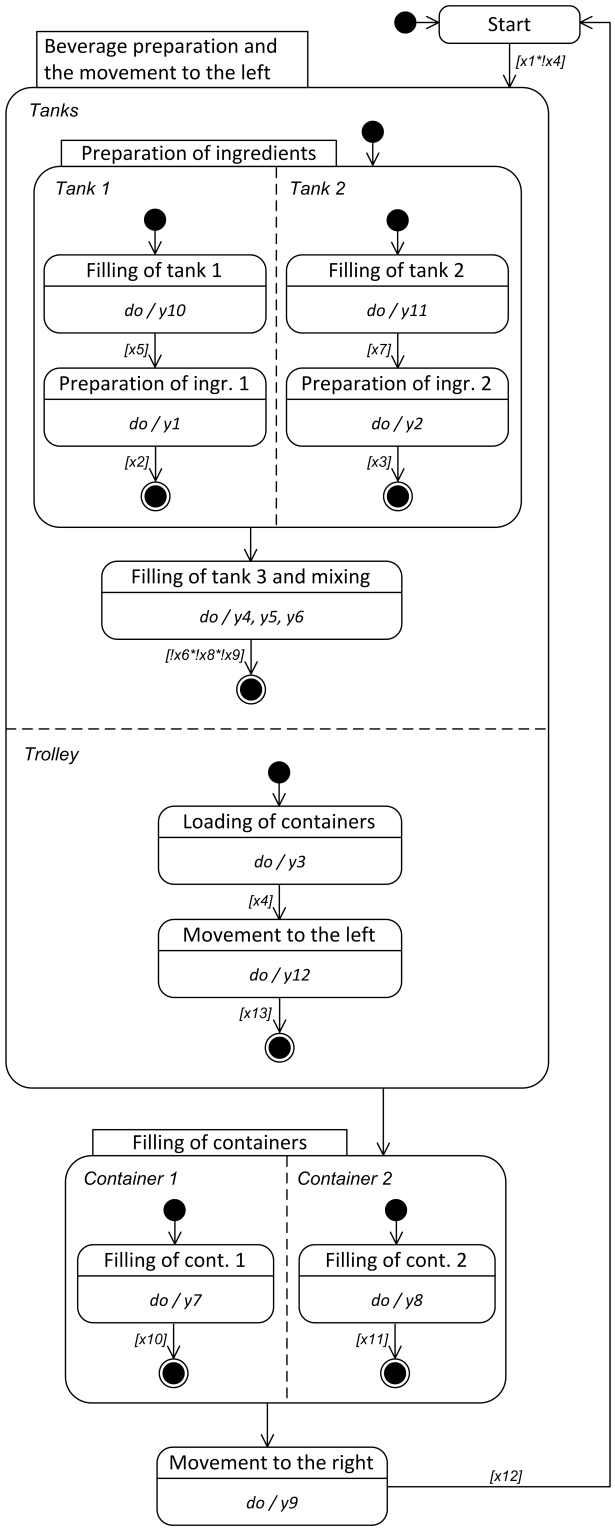

Fig. 5: A state machine diagram - the lowest hierarchy level.

6. Translation UML into HDL

The graphical specification of the designed system can be transformed and detailed to another specification. The proposed method is based on [11], [12]. Other important sources of inspiration were [13], [14] that use the Model-Driven Development (MDD) approach in the embedded systems design area. The proposed method used the above methods and adapted them to the Model Driven Architecture (MDA) on the model transformation level. The transformation path begins with UML state machine diagram then follows to a temporary HCFSM model and finally to a Verilog description. Moreover, modeling is limited to the modular diagrams, so there are no transitions between states on different levels of hierarchy (no transitions crossing borders of the states). Formal transformation rules were defined in QVT (Query/View/Transformation) language [15]. These rules describe how to transform a UML state machine model to an HCFSM model of the metamodel level.

The proposed method consists of several stages. In the first step, the state machine model (expressed as a UML diagram) is divided into a number of FSMs. In the next step, the FSM based on the state machine at the highest level of the hierarchy is chosen. It will play a primary (coordinating) function with regards to all the remaining FSMs. The next step is to augment the master FSM with additional signals associated with the activity of each of the sub-automata. Moreover, if the diagram contains completion transitions, relevant transition guards are modified by adding special signals using completion events (Fig. 6).

Next, the idle states have to be added in each subordinate automaton. The control is transferred to the idle state when the automaton is inactive (its activation is managed by the superior FSM). Moreover, it is also necessary to supplement the FSM with additional transitions to the idle state. The selected subordinate FSMs are shown in Fig. 7, Fig. 8 and Fig. 9.

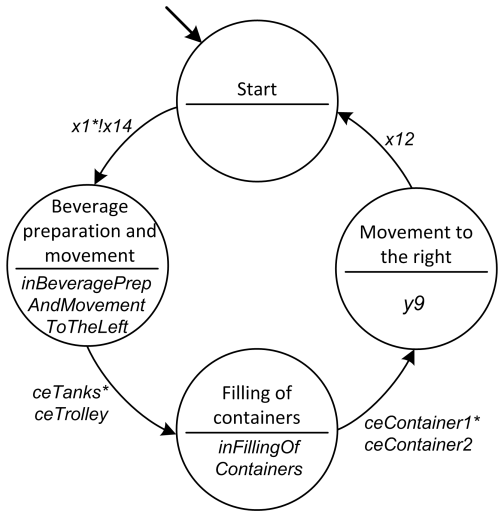

Fig. 6: A master FSM with added signals.

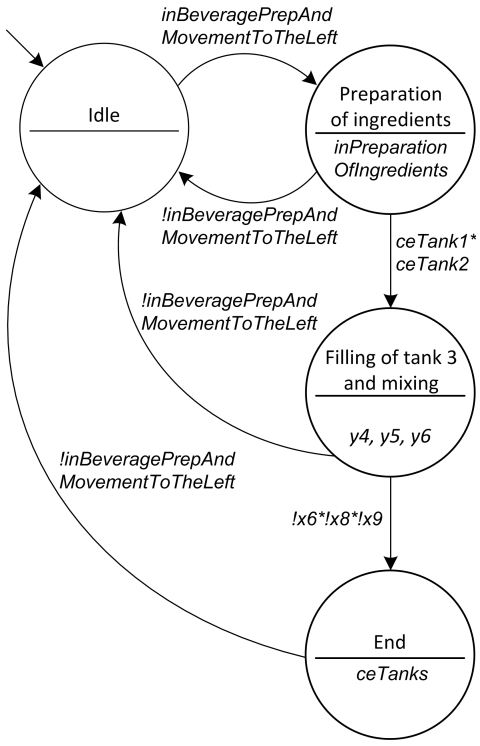

Fig. 7: The subordinate FSM Tanks with additional transitions and idle state.

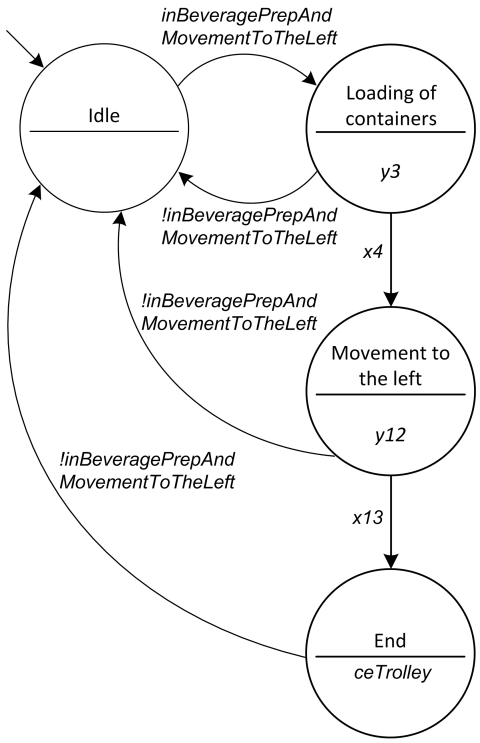

Fig. 8: The subordinate FSM Trolley with additional transitions and idle state.

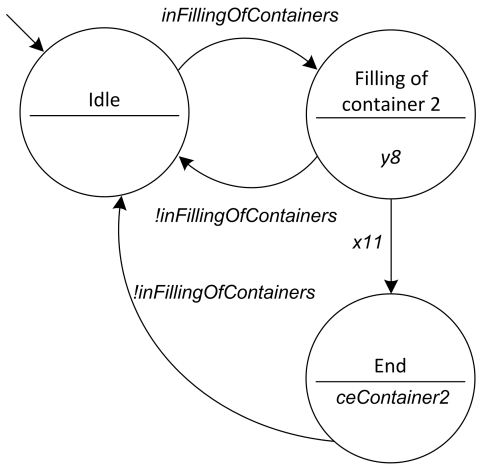

Fig. 9: The subordinate FSM Container2 with additional transitions and idle state.

The HCFSM diagram for the mixer example is shown in Fig. 10.

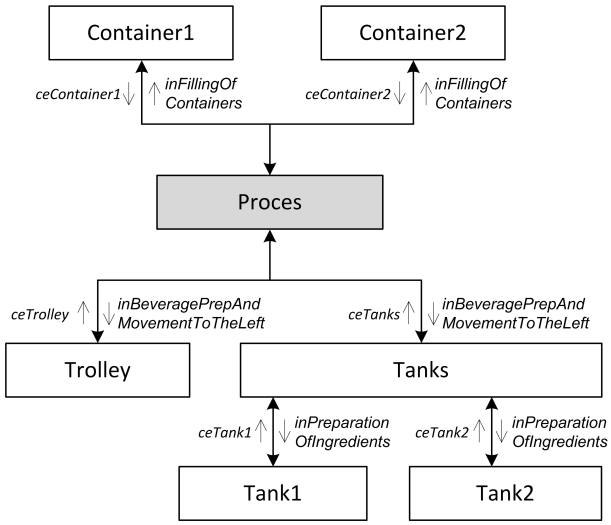

Fig. 10: A HCFSM diagram for the mixer example.

The final stage of the method is to generate Verilog files that describe the various decomposed FSM automata. There are many possibilities of FSM implementation in hardware description languages [16]. The behavioral description using one, two or three processes is one of the most popular ways, however it seems that the most readable approach is the version with two processes.

In Fig. 11 the master module "Top" in Verilog is presented.

7. U2V System

The developed methods and algorithms are implemented in the original CAD system called U2V [4].

```
module Top(clk,reset,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,
           y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12);

input clk,reset,x1,x2,x3,x4,x5,x6,x7,x8, x9,x10,x11,x12,x13;
wire inBeveragePrepAndMovementToTheLeft,
     inFillingOfContainers,ceTanks,ceTrolley,
     ceContainer1,ceContainer2;
output y1,y2,y3,y4,y5,y6,y7,y8,y9,y10, y11,y12;

Process FSM_PROCESS(clk,reset,x1,x4,x12,y9,
                    inBeveragePrepAndMovementToTheLeft,
                    inFillingOfContainers,ceTanks,ceTrolley,
                    ceContainer1,ceContainer2);

Tanks FSM_TANKS(clk,reset,x6,x8,x9,y4,y5,y6,
                inBeveragePrepAndMovementToTheLeft,
                inPreparationOfIngredients,ceTank1,
                ceTank2,ceTanks);

Trolley FSM_TROLLEY(clk,reset,x4,x13,y3,y12,
                    inBeveragePrepAndMovementToTheLeft,
                    ceTrolley);

Tank1 FSM_TANK1(clk,reset,x2,x5,y1,y10,
                inPreparationOfIngredients,ceTank1);

Tank2 FSM_TANK2(clk,reset,x3,x7,y2,y11,
                inPreparationOfIngredients,ceTank2);

Container1 FSM_CONTAINER1(clk,reset,x10,y7,
                          inFillingOfContainers,ceContainer1);

Container2 FSM_CONTAINER2(clk,reset,x11,y8,
                          inFillingOfContainers,ceContainer2);

endmodule
```

Fig. 11: Top module in Verilog.

The U2V input is specified as a text-file containing the description of the UML state machine model in XML format compliant with the XML Metadata Interchange (XMI) specification ver. 2.1. The UML model loaded into the program can be visually presented in the form of a tree structure of an XML document. Then the system analyzes the given XML file and creates a temporary HCFSM model using the transformation rules (expressed in QVT language). Next the HCFSM model is translated into a Verilog description (in a behavioral, synthesizable and modular form). Depending on the model complexity (the number of hierarchy levels), the generated specification can be written in one or more text files. The division of the output specification into a number of files increases its visibility and allows for independent simulation and verification of the model parts. Figure 12 shows the structure of the U2V system.

U2V was implemented in Java in the environment "Eclipse" with the plug-in "Jamon" [17], which provides the ability to generate text specifications based on created templates.

8. FPGA Implementation

The Verilog specification of Mixer was simulated in Aldec's Active-HDL system. The simulation results confirm that the designed system behavior corresponds to the specification expressed as a UML state machine diagram. Then in the Xilinx ISE environment the synthesis and implementation of the designed system was done. The results are presented in Tab. 1.

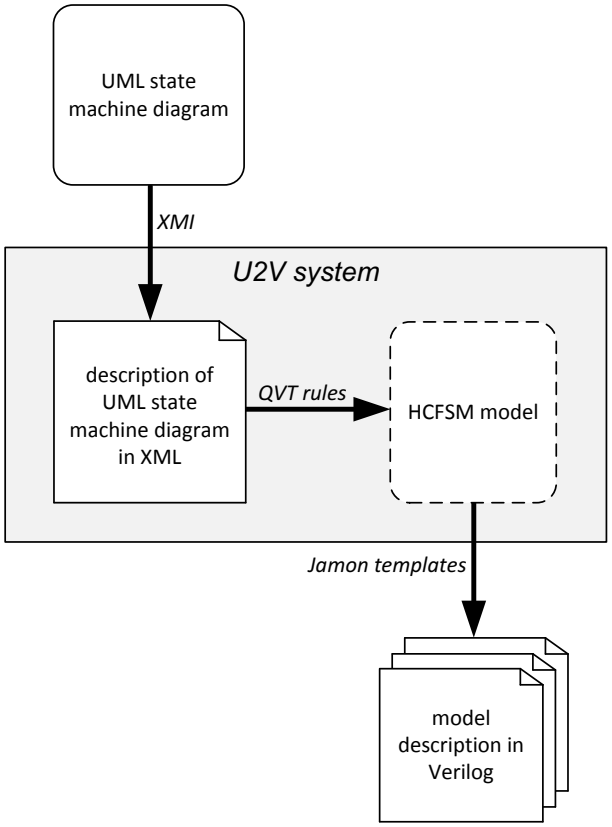

Fig. 12: The structure of U2V system.

Tab. 1: Results of synthesis and implementation of mixer example for the selected Xilinx devices.

	Spartan3 xc3s50	**Virtex4** xc4vlx1	**Virtex5** xc5vlx30
Number of flip flops	14	14	14
Number of LUTs	42	43	33
Number of slices	23	22	17
Number of IOBs	27	27	27

9. Conclusion

The graphical specification method for digital systems, especially logic controllers, was presented. The method is based on the UML state machine model, and with the help of a temporary HCFSM model, a behavioral, synthesizable and modular system description in a hardware description language (Verilog) is generated. The transformation on the metamodel level from UML into Verilog is made by using the MDD approach. The rules of transformation between UML model and temporary HCFSM model were defined in QVT language. The generated behavioral description in Verilog can afterwards be synthesized and implemented into FPGA.

The method developed here obviously has some limitations, which indicates the most important directions of further research, including:

- Extending the modeling capabilities of the non-modular systems in which transitions can cross the state borders and another types of UML diagrams (e.g. activity diagram, use-case diagram).
- Implementation in U2V system the verification methods for the UML diagrams.
- Optimization the FSMs implementation in digital circuits [18], [19].
- Developing translation methods from UML models into other formats, e.g. SystemC, SFC [12] and PNSF [1].

References

[1] WEGRZYN, M. Implementation of Safety Critical Logic Controller by Means of FPGA. *Annual Reviews in Control*. 2003. vol. 27, iss. 1, pp. 55–61. ISSN 1367-5788. DOI: 10.1016/S1367-5788(03)00007-5.

[2] LUBA, T. *Programmable Devices for Processing of Signal and Information*. Warsaw: WKL, 2008. 978-83-206-1711-5.

[3] WISNIEWSKI, R., A. BARKALOV, L. TITARENKO and W. A. HALANG. Design of microprogrammed controllers to be implemented in FPGAs. *International Journal of Applied Mathematics and Computer Science*. 2011, vol. 21, no. 2, pp. 401–412. ISSN 2083-8492. DOI: 10.2478/v10006-011-0030-1.

[4] BAZYDLO, G. and M. ADAMSKI. The specification of the hierarchical state machine from UML 2.4 and their automatic implementation in Verilog. *Przeglad Elektrotechniczny*. 2011, vol. 2011, no. 11, pp. 145–149. ISSN 0033-2097.

[5] VALETTE, R. Etude comparative de deux outils de representation: Grafcet et reseau de Petri. *Le Nouvel Automatisme*. 1978, vol. 7, iss. 3, pp. 377–382. ISSN 0220-8482.

[6] BOOCH, G., J. RUMBAUGH and I. JACOBSON. *The Unified Modeling Language User Guide*. 2nd ed. Indianapolis: Addison-Wesley Professional, 2005. ISBN 978-0321267979.

[7] VOGEL-HEUSER, B., S. BRAUN, B. KORMANN and D. FRIEDRICH. Implementation and evaluation of UML as modeling notation in object oriented software engineering for machine and plant automation. In: *Proceeding of the 18th IFAC World Congress*. Milan: International Federation of Automatic Control, 2011, pp. 9151–9157. ISBN 978-3-902661-93-7.

[8] Documents associated with Unified Modeling Language (UML): v2.4.1. In: *OMG: Object Management Group* [online]. 2011. Available at: www.omg.org/spec/UML/2.4.1.

[9] PILONE, D. and N. PITMAN. *UML 2.0 in a Nutshell*. 2nd ed. Farnham: O'Reilly Media, 2005. ISBN 978-0-596-00795-9.

[10] HAREL, D. Statecharts: A visual formalism for complex systems. *Science of computer programming*. 1987, vol. 8, no. 3, pp. 231–274. ISSN 0167-6423. DOI: 10.1016/0167-6423(87)90035-9.

[11] DRUSINSKY, D. and D. HAREL. Using statecharts for hardware description and synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 1989, vol. 8, no. 7, pp. 798–807. ISSN 0278-0070. DOI: 10.1109/43.31537.

[12] ADAMSKI, M. Petri nets in ASIC design. *Applied Mathematics and Computer Science*. 1993, vol. 3, no. 1, pp. 169–179. ISSN 2083-8492.

[13] WOOD, S., D. AKEHURST, O. UZENKOV, W. HOWELLS and K. MCDONALD-MAIER. A model-driven development approach to mapping UML state diagrams to synthesizable VHDL. *IEEE Transactions on Computers*. 2008, vol. 57, no. 10, pp. 1357–1371. ISSN 0018-9340. DOI: 10.1109/TC.2008.123.

[14] LU, S., W. A. HALANG and L. ZHANG. A component-based UML profile to model embedded real-time systems designed by the MDA approach. In: *Proceeding of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. Hong Kong: IEEE, 2005, pp. 563–566. ISBN 0-7695-2346-3. DOI: 10.1109/RTCSA.2005.6.

[15] Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT): ver. 1.1 Beta 2. In: *OMG: Object Management Group* [online]. 2009. Available at: http://www.omg.org/spec/QVT/1.1/Beta2/PDF/.

[16] XST User Guide for Virtex-4, Virtex-5, Spartan-3, and Newer CPLD Devices: UG627 (v 12.4). In: *XILINX* [online]. 2010. Available at: www.xilinx.com.

[17] Jamon - a typed template engine for Java. In: *JAMON* [online]. 2011. Available at: www.jamon.org.

[18] RAWSKI, M., H. SELVARAJ and T. LUBA. An application of functional decomposition in ROM-based FSM implementation in FPGA devices. *Journal of Systems Architecture*. 2005, vol. 51, iss. 6–7, pp. 424–434. ISSN 1383-7621. DOI: 10.1016/j.sysarc.2004.07.004.

[19] BARKALOV, A., L. TITARENKO and S. CHMIELEWSKI. Decrease of hardware amount in logic circuit of Moore FSM. *Telecommunication Review and Telecommunication News*. 2008, vol. 81, no. 6, pp. 750–752. ISSN 1230-3496.

About Authors

Grzegorz BAZYDLO received his Ph.D. from the University of Zielona Gora (Poland) in 2010. Currently he is an assistant professor at the Institute of Computer Engineering and Electronics. His current research interests include the graphical methods (especially UML) in design, synthesis and verification of discrete systems.

Marian ADAMSKI received his Ph.D. from Silesian Technical University (Gliwice, Poland), in 1976. Currently he is Full-Professor at the Institute of Computer Engineering and Electronics. His research interests include mathematical logic and Petri nets in digital systems design, formal development of Logic Controller programs, VHDL, FPLD and FPGA in industrial applications.

Marek WEGRZYN received his Ph.D. from the Warsaw University of Technology (Poland) in 1999. Currently he is Head of the Computer Engineering Division. His research interests include digital systems design, Hardware Description Languages (HDLs), Petri nets, concurrent controller designs, programmable logic, and information technology in industrial applications.

Alfredo ROSADO MUNOZ received his Ph.D. from the University of Valencia (Spain) in 2000. Currently he is Associate Professor at the Department of Electronic Engineering. His research interests include signal processing algorithm implementation, embedded systems design, FPGA, and VHDL.