DOI: 10.31114/2078-7707-2020-1-8-14

Application of visual tools for system modeling of digital integrated circuits

S.E. Khalzev, A.I. Vlasov, V.A. Shakhnov

Bauman Moscow State Technical University shakhnov@iu4.bmstu.ru

***Abstract* — This paper is devoted to the analysis of use of the xtUML diagram language for system level design of digital ICs. Design levels of digital ICs (system, RTL, physical) are briefly considered. The features of the xtUML diagram language for system level design are analyzed. The methodology of using the xtUML diagram language for system design of digital ICs with subsequent modeling and translation into a lower design level is proposed.**

***Keywords* — system design, IC, VLSI, xtUML, diagram language, design flow.**

I. Introduction

A trend towards miniaturization has emerged in the electronics industry. The culmination of miniaturization today is the system-on-chip – an integrated circuit (IC) containing very large functional units of the device. Such ICs are the basis for building mobile phones, computers, etc. Often systems-on-a-chip are so complex that for the successful implementation of such an IC project, it is necessary to organize its hierarchical decomposition – to single out simple components.

At the initial stage of designing digital IC, a system model is developed that includes a behavioral description and the environment of the developed system, which allows reflecting the interaction of IC with other elements of equipment or measurement objects. This approach is effective in the development of modern single-chip systems of high complexity. And if we draw analogies with concepts from a Russian Unified System of Engineering Drawings (ESKD), then as a result of system design, the engineer should get some kind of structural diagram of the product being developed. However, with a further transition from the system level to the level lower in the hierarchical decomposition, the problem arises of the complexity of the automated interpretation and formalization of the "rectangles" of the structural diagram in such a way that would be perceived by electronic CAD systems of the next, low-level design stages. This type in modern design routes for the digital element base can be a behavioral description of the system in the C, MATLAB, SystemC programming languages, etc. [1].

Attempts to solve the aforementioned problem led to the introduction of a high-level software tool that was previously used either with traditional computing systems or was aimed at solving completely different problems, the xtUML language, which is an executable extension of the classical UML diagram language, into the traditional design route.

II. Analysis of a typical route for designing a digital IC

In the process of developing IC, various levels of abstraction are identified as part of the design flow. So, Fig. 1 shows a typical digital IC design flow. It can be seen from it that, depending on the level of representation, the object of abstraction is the system, register, gate, geometry of the library element on the crystal [2], [5]-[7].

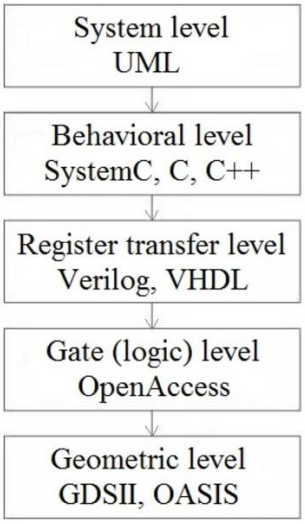

Fig. 1. Typical digital IC design flow

The system level is a description of the system at the level of the structural diagram in the diagram language. The behavioral level of the project description consists of the behavioral description of the system in terms of functions, expressions, algorithms. The register transfer level is a combination of arithmetic and logical nodes, memory elements, etc. The gate (logical) level describes the project at the level of logic gates and triggers. The lowest is the geometric level. On it, logical elements are represented at the layout level in the form of geometric elements and interconnections.

From the analysis of the design route of the digital IC, it can be seen that as the descent through the hierarchical decomposition levels during the design process, the digital IC developer has to deal with less abstract and more specific and complex structures with each new stage. Thus,

the synthesis of new and debugging of current solutions is easier and more profitable at the system level.

III. XTUML LANGUAGE ANALYSIS AND BASIC DIAGRAM LANGUAGE ELEMENTS

To build system models, IC developers can use the UML extension xtUML (eXecutable and Translatable Unified Modeling Language).

The xtUML language has the following features:

1) it's a diagram language that combines all the previously known features of the UML language with the new features of transient diagram modeling;

2) diagrams described in xtUML language are platform independent, they can be launched, tested and debugged without the need to generate any code;

3) xtUML diagrams can be easily translated into SystemC, C or C++ code using the model compiler;

4) The BridgePoint design environment from Mentor Graphics, which developed the xtUML diagram language, is free and available for download at https://xtuml.org.

The xtUML diagram language can be used simultaneously for software development and as a highly abstract approach to the design of electronic equipment. Since the language is platform independent, it opens up wide opportunities for system engineers to reuse of universal diagrams from previous projects in new systems without changing the target hardware [3].

While the classic UML language includes the main models – use case, activity, placement, deployment, and logical, xtUML language contains only a domain model. A domain is an entity that is an analog of a block in block diagram of a system. The domain model incorporates the following diagrams:

1) domain diagram – shows model domains and dependencies between them,

2) class diagram – shows the model classes and their relationships,

3) state diagram – shows class states, events and state transitions.

The mindmap of the domain model diagrams is shown in Fig. 2.

xtUML conceptual modeling is performed for each domain in the system. To determine the domains, the system engineer must collect information on the use case for each domain. Dependencies between domains, the analogues of which in the electrical block diagrams show the signal propagation paths, are called bridges.

Thus, the domain diagram implements the highest level of system abstraction. Creating a domain diagram can take only a few hours, this diagram can be updated several times during the development cycle of the system as the interfaces of "communication" between domains are refined.

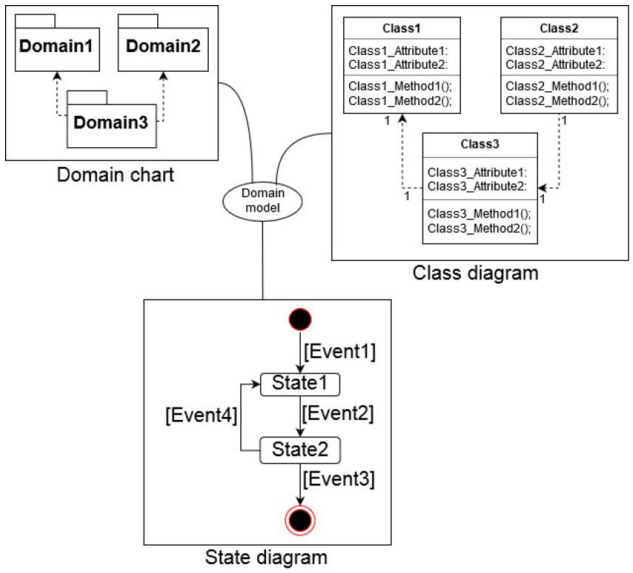

Fig. 2. The mindmap of the domain model diagrams of xtUML

A domain is both an abstract and a specific entity. For example, the domain "Signal Generator" may not specify the internal structure of the signal generator, but its input and output characteristics, such as the frequency and amplitude of the input and output signals, cannot be vague and indefinite. An example of a domain diagram is shown in Fig. 3.

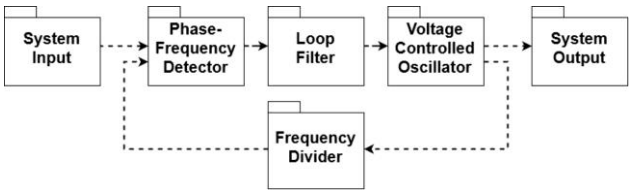

Fig. 3. Domain diagram for an example of a frequency synthesizer

Domains are filled with real and abstract entities. These entities, both real and abstract, are called classes. Several classes with links form a class diagram.

For classes, their attributes are assigned – a set of properties that characterize the operation of a given class. Each attribute shows the characteristic of the signal the class is working with. By connecting the attributes of different classes among themselves, an association occurs between the classes themselves. Class methods are functions that are available to the class when working with a signal. Associations show the propagation of signals. The multiplicity of relations shows how many instances of one class can relate to instances of another class [4]. An example of class diagram is shown in Fig. 4.

Many things have life cycles – a set of states that a thing goes through during its work. Therefore, the life cycle can be shown in the form of a state diagram.

There is always a circle on the state diagram that indicates the initial state, as well as a circle with a circle inside, indicating the final state. The states in the diagram are indicated by a rectangle with rounded corners. The

states are connected by arrows indicating the transition from one state to another and showing its direction [4]. An example of a state diagram is shown in Fig. 5.

Each state in the state diagram has an associated procedure that takes as input data elements associated with the event that triggered the entry into the state. Each procedure contains a set of actions, and each action can perform some functional calculation, data access, signal generation, etc. Actions are similar to code, with the exception of a higher level of abstraction, which makes no assumptions about the structure of the software or its implementation. Some language constructs are presented in Table 1.

An operation procedure of a frequency-phase detector of frequency synthesizer is presented in Table 2.

Thus, it can be seen that with help of diagrams of the xtUML language, one can completely describe the structure and algorithm of the system.

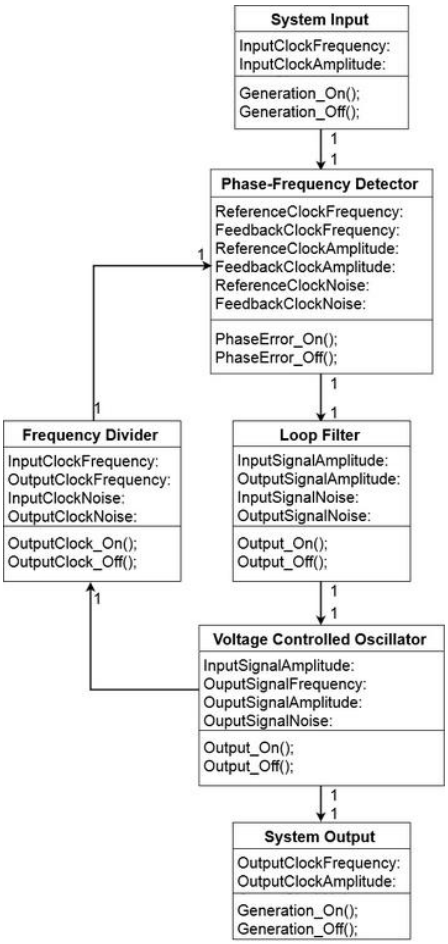

Fig. 4. Class diagram for an example of a frequency synthesizer

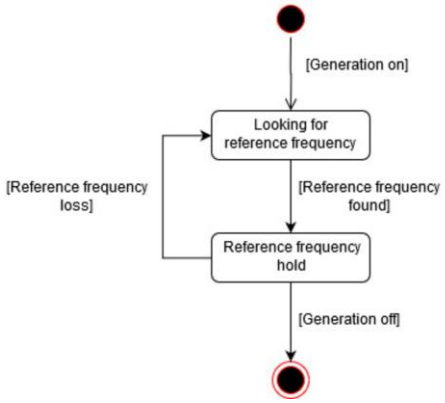

Fig. 5. State diagram for an example of a frequency synthesizer

Table 1

Listing of xtUML language constructs

```
//Comment

if (<boolean expression>) //If boolean expression is true,
<statement> //statement is executed
elif (<boolean expression >) //If current boolean
expression is true and previous is false,
<statement> //statement is executed
else //If all boolean expressions are false,
<statement> //statement is executed
end if;

for each <entity instance> //assigning a statement
in <set of entities> //to a set of entities
<statement>
end for;

while <boolean expression> //While boolean expression
is true,
<statement> //statement is executed
end while;

break; //Interruption of a cycle

continue; //Resuming the cycle
```

Table 2

Operation procedure of a frequency-phase detector of frequency synthesizer

```
if (state.generate == 1) //If the generation process is
running
    if(f_sys<f_ref - 50) //If the frequency drops more
    than 50 Hz
        f_sys_up(); //increase the frequency
    if(f_sys>f_ref + 50) //If the frequency has increased
    by more than 50 Hz
        f_sys_dn(); //drop the frequency
endif;
```

IV. METHODOLOGY FOR BUILDING SYSTEM MODELS USING THE XTUML LANGUAGE

When building system models in the xtUML language, there are three stages of design:

1) preparatory stage,

2) development stage,

3) stage of implementation of the model.

The preparatory stage consists of the following actions:

1) Clarification of the technical requirements for the system and their synchronization with the use cases.

2) Partitioning the system into domains.

3) Identification of new and reusable domains.

4) Assessment of the key risks of the system design.

5) Assessment of the volume of work and its cost.

At the development stage, iterative design takes place, which includes the following stages:

1) Updating the system domain diagram (if necessary).

2) Creating a class diagram of the system.

3) Creating a state diagram of the system.

4) Adding operations to classes.

5) Indication of operations and action states in classes using procedures.

6) Indication of initial conditions and test methods using procedures.

7) Bridging between domains.

8) Simulation of the resulting scenario.

Upon completion of the development stage, the engineer receives an abstract model of his system.

The stage of model implementation includes only one action, which consists in synthesizing the system code using a compiler for a particular programming language.

As a result of applying the above methodology, an engineer gets a well-functioning high-level model of the developed system, suitable for further designing the digital ICs at lower levels – obtaining a description of the system in low-level languages. The ways of further using the simulation results in the xtUML language are presented in Fig. 6 [8], [9].

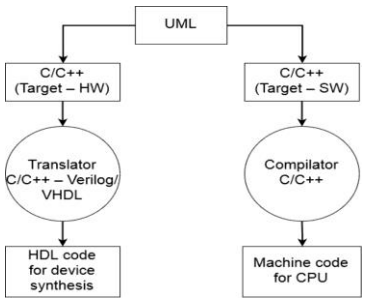

Fig. 6. Ways to use xtUML simulation results

The xtUML model generates two C/C++ codes. The first of them is converted into machine code of the already configured processor core logic. The second is converted into a Verilog/VHDL description of programmable system logic.

Summing up the analysis of the methodology for using the xtUML language at the initial stages of the design flow of the digital ICs, it can be noted that this methodology fully utilizes the capabilities of the xtUML diagram language, and when it is used, the probability of errors at the system design stage is minimized.

V. EXAMPLE OF THE DEVELOPMENT OF A SYSTEM MODEL USING THE XTUML LANGUAGE

As an example, we will develop xtUML diagrams for dynamic random access memory (DRAM) IC. The ultimate goal of creating a system model in the xtUML diagram language is code synthesis that can be used at lower levels of system design.

Begin work by analyzing the block diagram of the system being developed. The block diagram of DRAM IC is shown in Fig. 7.

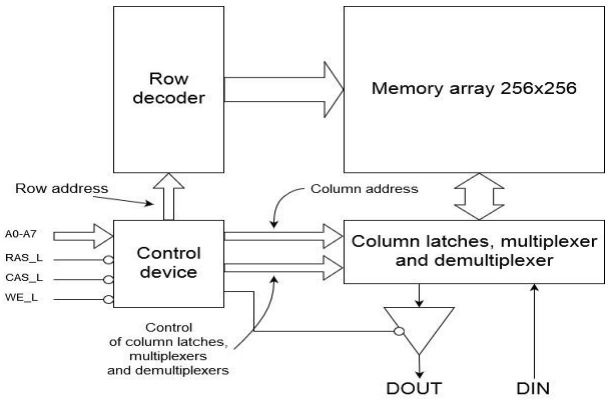

Fig. 7. Block diagram of DRAM IC

In order to understand what functions of the system are available to each of its users, we will draw up a use case diagram. A use case diagram of DRAM IC is presented in Fig. 8.

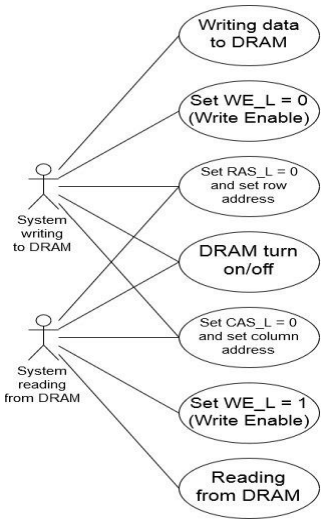

Fig. 8. Use case diagram of DRAM IC

It can be seen from the use case diagram that DRAM IC has two use cases: a process when some external system writes data to DRAM and a process when some system reads data from DRAM.

In accordance with the block diagram of the system depicted in Fig. 7, we will divide the system into domains: each block in the block diagram must have its own domain. The domain diagram of DRAM IC system model is shown in Fig. 9.

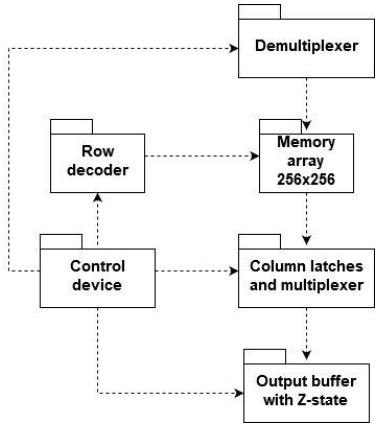

Fig. 9. Domain diagram of DRAM IC system model

The domain diagram decomposes the system – splits the original system into several elementary subsystems.

The creation of the domain diagram completes the first stage of the development of the system model – the preparatory stage. The next is the development stage. Its key steps are:

1) Creating a class diagram of the system.
2) Creating a state diagram of the system.
3) Adding operations to classes.
4) Indication of operations and action states in classes using procedures.
5) Indication of initial conditions and test methods using procedures.
6) Simulation of the resulting scenario.

The set of attributes and methods for the class diagram is made up of the functionality that is required from the developed DRAM IC. In this case, the class attributes are the frequency of information signals, the width of the data bus. Methods are functions such as "enter the row address", "enter the column address", "enable data writing", "enable data reading ". The class diagram of the DRAM IC system model is presented in Fig. 10.

From the class diagram it becomes clear what signals each class works with and what functions each of them performs.

A set of states and transitions for a state diagram is compiled based on the operating modes of the DRAM IC. The operating modes of DRAM IC can be traced by the time diagrams of writing and reading. Timing diagrams of writing and reading data for DRAM IC are presented in Fig. 11 and Fig. 12.

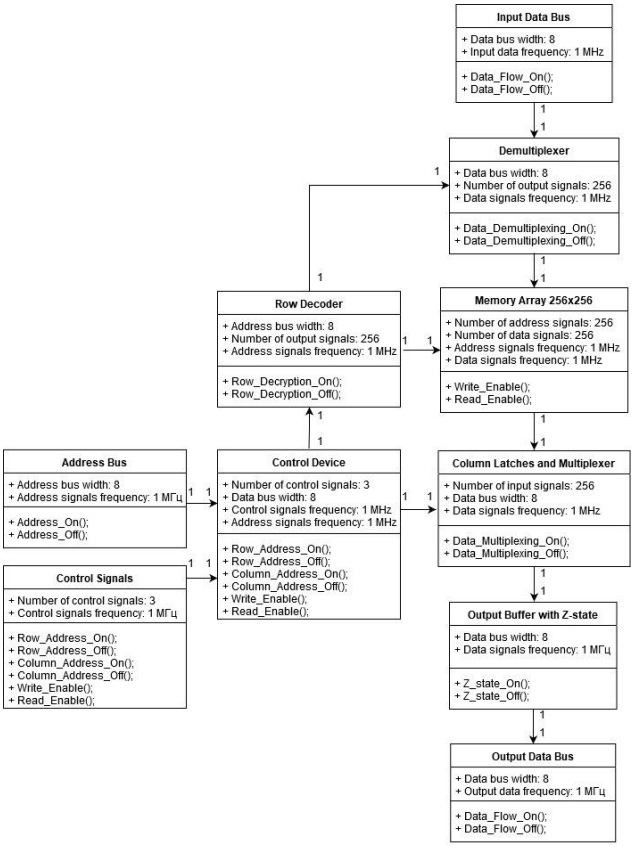

Fig. 10. Class diagram of the DRAM IC system model

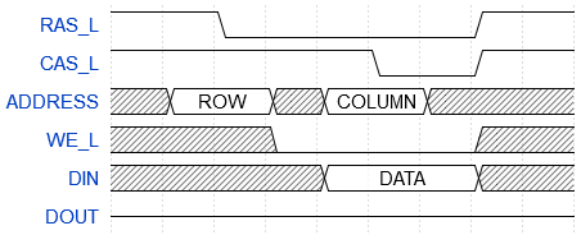

Fig. 11. Timing diagram of writing data to DRAM IC

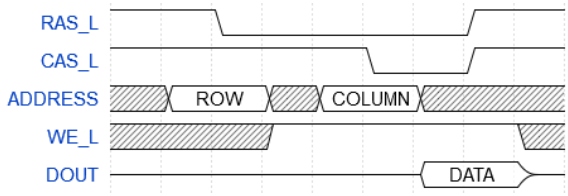

Fig. 12. Timing diagram of reading data from DRAM IC

DRAM states of dynamic memory are “Waiting for commands”, “Set a row address”, “Set a column address”, “Writing data to memory”, “Reading data from memory”. The DRAM IC transitions are “IC enable”, “Arrival of the Read_Enable signal”, “Arrival of the Write Enable signal”, “Start of data reading”, “Start of data recording”, “End of data reading”, “End of data recording”. The state diagram of the DRAM IC system model is presented in Fig. 13.

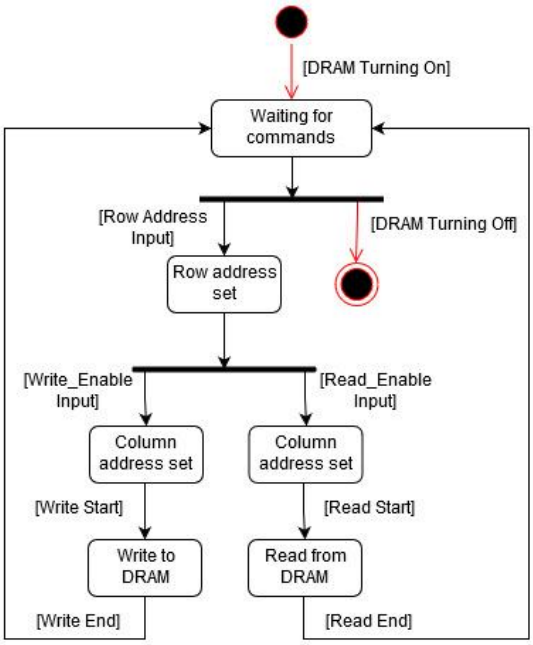

Fig. 13. State diagram of the DRAM IC system model

The state diagram of the IC model provides the basis for the synthesis of the finite state machine of the system.

Diagrams of domains, classes and states give complete information about the system being developed. Further, after describing the diagrams system engineer can simulate and debug the system in Mentor Graphics BridgePoint, as well as synthesize its structure in the hardware description language. This step allows the system developer, who used the technique described in section IV, to obtain a starting point for the design of the system at a low level.

VI. Conclusion

Проектирование на системном уровне является первым этапом в маршруте проектирования цифровых СБИС. Для работы на этом уровне язык диаграмм xtUML предоставляет возможности симуляции и верификации проектов, а также даёт системному инженеру базу для дальнейшего низкоуровневого проектирования СБИС.

Design at the system level is the first step in the design path of digital IC. To work at this level, the xtUML diagram language provides simulation and verification capabilities for projects, and also provides the system engineer with the basis for further low-level IC design.

Support

Separate research results were obtained within the framework of the state task of the Ministry of Education and Science of the Russian Federation “Research on methods and models for the synthesis of energy-efficient inorganic memristor structures”.

References

[1] Merkutov Lohov A. and Rabovolyuk A. 2007. Elektronika: Nauka, Tekhnologiya, Biznes. 3 p. 102-109.

[2] V.B. Steshenko, A.V. Rutkevich, E. Gladkova and others. Proektirovanie SBIS tipa «Sistema na kristalle». Marshrut proektirovaniya. Sintez sxemy. Chast 1 (Designing of VLSI type "System on a chip". The design route. The synthesis scheme. Part 1) // Elektronnye komponenty. – 2009. - №1. – p. 14 –21.

[3] Stephen J. Mellor, Marc J. Balcer, Executable UML: A Foundation for Model-Driven Architecture // Addison-Wesley Professional. 2001. 161 p.

[4] Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide Second Edition. Addison Wesley Professional, 2005. 496 p. (Russ. ed.: Buch G., Rambo D., YAkobson I. YAzyk UML. Rukovodstvo pol'zovatelya. 2-e izd.: Per. s angl. Muhin N. – Moscow, DMK Press Publ., 2006. 496 p.).

[5] Demin A.A., Vlasov A.I. Application of Hexagonal Conceptual Model for Solving Problem of Synchronization by Visual Designing of Complex Systems // Breakthrough directions of Scientific Research in NRNU MEPhI: Development Perspectives in the Framework of the Strategic Cep."KnE-Engineering" 2018. C. 266-273.

[6] A. Rabovolyuk. Obzor marshruta proektirovaniya PLIS FPGA Advantage kompanii Mentor Graphics (Mentor Graphics FPGA Advantage FPGA design route overview) // Komponenty i tekhnologii. 2005. №5. p. 98-101.

[7] A.I. Vlasov, A.A. Karpunin, IU.M. Ganev. Sistemnyi podkhod k proektirovaniiu pri kaskadnoi i iterativnoi modeli zhiznennogo tsikla (A systematic approach to design with a cascading and iterative life cycle model) // Trudy mezhdunarodnogo simpoziuma “Nadyozhnost i kachestvo”. 2015. T. 1. p. 96-100.

[8] A.I. Vlasov. Prostranstvennaia model otsenki evoliutsii metodov vizualnogo proektirovaniya slozhnykh sistem (Spatial model for assessing the evolution of visual design methods for complex systems) // Datchiki I sistemy. 2013. №9 (172). p. 10-28.

[9] A.I. Vlasov. Konceptsyya vizualnogo analiza slozhnykh sistem v usloviyakh sinkhronnykh tekhnologii proektirovaniya (The concept of visual analysis of complex systems in the conditions of synchronous design technologies) // Datchiki I sistemy. 2016. № 8-9(206). p. 19-25.

Применение визуальных средств для системного моделирования цифровых интегральных схем

С.Е. Хальзев, А.И. Власов, В.А. Шахнов

Московский Государственный Технический Университет им. Н. Э. Баумана,
shakhnov@iu4.bmstu.ru

***Аннотация* — Работа посвящена анализу применения языка диаграмм xtUML на системном уровне проектирования цифровой элементной базы. Кратко рассмотрены уровни проектирования цифровой элементной базы (системный, регистровых передач, физический). Проанализированы особенности языка xtUML для системного уровня проекта. Предложена методика применения языка xtUML для системного проектирования цифровых интегральных схем с последующим моделированием и трансляцией в более низкий уровень представления проекта.**

***Ключевые слова* — системное проектирование, СБИС, xtUML, язык диаграмм, маршрут проектирования.**

Литература

[1] Lohov A. and Rabovolyuk A. 2007. Elektronika: Nauka, Tekhnologiya, Biznes. 3 p. 102-109

[2] В.Б. Стешенко, А.В. Руткевич, Е. Гладкова и др. Проектирование СБИС типа «Система на кристалле». Маршрут проектирования. Синтез схемы. Часть 1 // Электронные компоненты. – 2009. - №1. – С. 14 –21.

[3] Stephen J. Mellor, Marc J. Balcer, Executable UML: A Foundation for Model-Driven Architecture // Addison-Wesley Professional. 2001. 161 p.

[4] Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс. – 496 с.: ил. ISBN 5-94074-334-X

[5] Demin A.A., Vlasov A.I. Application of Hexagonal Conceptual Model for Solving Problem of Synchronization by Visual Designing of Complex Systems // Breakthrough directions of Scientific Research in NRNU MEPhI: Development Perspectives in the Framework of the Strategic Cep."KnE-Engineering" 2018. С. 266-273.

[6] Рабоволюк А. Обзор маршрута проектирования ПЛИС FPGA Advantage компании Mentor Graphics // Компоненты и технологии. 2005. №5. С. 98-101.

[7] Власов А.И., Карпунин А.А., Ганев Ю.М. Системный подход к проектированию при каскадной и итеративной модели жизненного цикла // Труды международного симпозиума «Надежность и качество». 2015. Т. 1. С.96-100.

[8] Власов А.И. Пространственная модель оценки эволюции методов визуального проектирования сложных систем // Датчики и системы. 2013. №9 (172). С. 10-28.

[9] Власов А.И. Концепция визуального анализа сложных систем в условиях синхронных технологий проектирования // Датчики и системы. 2016. № 8-9(206). С. 19-25.