Visual methods of high-level system design for digital hardware components

S E Khalzev, A I Vlasov and V A Shakhnov
Bauman Moscow State Technical University, 5, 2-ya Baumanskaya, Moscow, 105005, Russia

E-mail: shakhnov@iu4.bmstu.ru

Abstract. The article presents an analysis of using xtUML for high-level design of very large-scale integrated circuits. A design flow of digital VLSI circuits was investigated. The focus was made on the formalisation of the digital circuit models at the initial stages of system design. We studied specific features of developing a digital circuits’ system model that includes the behavioural description and the environment of the system being developed. This makes it possible to show the interaction between the very large-scale integrated circuits and other hardware components or measurement objects. However, further transition from the system level to the lower level in the hierarchical decomposition reveals the problem of “semantic gap”, meaning the complexity of the automated interpretation and formalisation of the conditional “rectangles” of the visual model to the form which could be interpreted by the tools used at the low-level design stages. The use of visual methods of system design is effective in the development of modern and highly sophisticated systems on a chip. We analysed xtUML diagrams at the system level of the digital very large-scale integrated circuit projects and formulated the procedure of using xtUML for developing digital VLSI circuits’ system models.

1. Introduction
In conditions of the industrial digitalisation, the trend for miniaturization of electronic components is becoming increasingly evident. The apogee of miniaturization today is a system on a chip, which is a very large-scale integrated circuit containing extra-large functional units of the device. Such very large-scale integrated (VLSI) circuits make up the basis of the mobile phones, computers, electronics, Internet of things, etc. It is a usual thing that due to the systems on a chip (SoCs) complexity, to perform the VLSI project it is necessary to make its hierarchical decomposition, that is to highlight its simple components [1].

The initial stage of the VLSI circuit design implies the development of a system model, which includes the behavioural description and environment of the system being developed. This makes it possible to show the interaction between the VLSI circuits and other hardware components or measurement objects. The approach is effective in the development of modern and highly sophisticated SoCs. If drawing analogies with the concepts from the unified system of design documentation, as a result of system design, the engineer should get some kind of block diagram of the product being developed. However, further transition from the system level to the lower level in the hierarchical decomposition reveals the problem of the complexity of the automated interpretation and formalisation of the conditional “rectangles” of the block diagram to the form which could be interpreted by the computer-aided design (CAD) systems used at the low-level design stages. In the modern design
flows of the digital hardware components, this form could be represented by the system behavioural description with C, MATLAB, SystemC and other languages [2].

Thanks to the attempts to solve this problem, the traditional design flow was complemented with a high-level visual tools, which were previously used either with traditional computing systems or were focused on solving completely different problems – the xtUML, which is an executable extension of the classical UML modelling language [3-5].

2. Analysis of a standard design flow for digital hardware components

According to the design flow, the developing of VLSI circuits consists of various levels of abstraction. Thus, figure 1 illustrates digital VLSI design flow. It shows that, depending of the presentation level, the abstraction is focused on the system, register, gate and geometry of the library cell on a chip [2].

![Figure 1. Standard design flow of digital hardware components.](image)

The system level is a system description at the block diagram level using the diagram language. The behavioural level of the project description comprises system behavioural description in terms of functions, expressions and algorithms. The register transfer level is a combination of arithmetic and logical nodes, memory cells, etc. The gate (logic) level describes the project at the level of logic gates and triggers. The geometric level is the lowest one. It represents the logic gates at the silicon (topological) level in the form of topographical features and interconnects.

The analysis of the digital hardware components design flow illustrates that the lower the design level in the hierarchical decomposition, the less abstract, vice versa, the more specific and complex structures represent the entities the developer has to work with. In this regard, the compilation of new and debugging of current solutions is easier and more feasible at the system level.

3. Analysis of xtUML and main language features

To build system models, VLSI developers can use the UML extension called xtUML (eXecutable and Translatable unified modelling language).

XtUML has the following features:

- it is a diagram language that combines all previously known features of the UML with new features of temporary diagram-based modelling;
- xtUML diagrams are platform independent, they can be run, tested and debugged without the need for any code generation;
- xtUML diagrams can be easily translated into SystemC, C or C++ code using a model compiler;
- BridgePoint environment by Mentor Graphics (the developer of xtUML) is free and available for download at https://xtuml.org.

XtUML can be used for software development and as a highly abstract approach to the electronic components design. Since the language is platform independent, it opens up wide opportunities for system engineers to utilise the reuse techniques, i.e. the reuse of universal diagrams from previous projects in new systems without changing the target hardware [6].

While the classic UML includes the basic models for use cases, activities, placement, deployment, and logic, the visual xtUML contains a domain model only. The domain is an entity that is an analogue of a block in a system block diagram. The domain model incorporates the following diagrams:

- domain chart to show model domains and dependencies between them,
- class diagram to show model classes and dependencies between them,
- state diagram to show class states, events, and state transitions.

Figure 2 presents a mental map of the domain model diagrams.

![Mental map of xtUML domain model diagrams](image)

**Figure 2.** Mental map of xtUML domain model diagrams.

As an example, below is the development of VLSI system model for USB–UART interface converter. This converter is designed to connect the devices that support the UART interface with a computer via a high-speed USB bus. Figure 3 illustrates a VLSI block diagram for USB–UART interface converter [7].

Figure 3 does not show the voltage stabilizer for the VLSI circuit of the USB–UART interface converter, since, being an analogue unit, is not intended for modelling in xtUML.

XtUML conceptual modelling is performed for each system domain. To define the domains, the system engineer has to collect information about use cases for each domain. Dependencies between domains, the analogues of which in the electrical diagrams show signal paths, are called bridges.

Thus, the domain chart implements the highest level of system abstraction. It takes a few hours to create a domain chart and it is possible to update is several times during the system development cycle as the interfaces of communication between domains are refined.
The domain is both an abstract and a specific entity. For example, the 'USB Controller' domain does not have to specify the structure of the USB controller interface but its input and output parameters, such as the interface speed and the width of the internal data bus, cannot be vague and undefined. Figure 4 shows an example of the domain chart.

The domains are filled with particular things and abstract entities. Both particular things and abstract entities are called classes. Several classes with associations form a class diagram.

The classes are characterised with attributes – a set of properties that characterize the work of this class. Each attribute shows the parameters of the signal the class is working with. The connection of the attributes of different classes makes up the association between the classes. The class methods are the functions that are available to the class when working with a signal. The associations show signal paths. The multiplicity of relations describes how many instances of one class can belong to instances of other class [8]. Figure 5 shows an example of the class diagram.

Many objects have life cycles, meaning a set of states that the objects pass through their work. For this reason, it is possible to show the life cycle in the form of a state diagram.

The state diagram always features a circle, which indicates the initial state, as well as a circle with another 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 the direction of such transition [8]. Figure 6 shows an example of the state diagram.

Each state in the state diagram has an associated procedure according to which the data elements associated with the event that triggered the entrance to the state are taken as the input data. Each procedure contains a set of actions, and each action can perform some functional calculation, data access, signal generation, etc.
Figure 5. Class diagram, the case of VLSI circuit of the USB–UART interface converter.

Figure 6. Class diagram, the case of VLSI circuit of the USB–UART interface converter.

The actions are similar to a code, with the exception of a higher level of abstraction, which makes
no assumptions about the structure of the software or its implementation. Figure 7 shows some of the language structures [6].

```plaintext
//Comment
if (<boolean expression>) //If boolean expression is true
    <operator> //execute operator
elif (<boolean expression>) //If current boolean expression is true, //while the previous one is false
    <operator> //execute operator
else //if all boolean expressions are false
    <operator> //execute operator
end if;

for each <instance> //assign operator in <set of instances> //to the set of instances
    <operator>
end for;

while <boolean expression> //While boolean expression is true,
    <operator> //execute operator
end while;

break; //Break the cycle
continue; //Continue the cycle
```

**Figure 7.** Listing of xtUML structures.

Figure 8 presents the procedure of UART package decoding in the case of VLSI circuit of the USB–UART interface converter.

```plaintext
if (Start_bit_set() == 1) //If the state bit is identified
    UART_package_decode(1); //Start UART package decoding
end if;

if (Stop_bit_set() == 1) //If the stop bit is identified
    UART_package_decode(0); //Stop the decoding //of UART package
end if;
```

**Figure 8.** Procedure of UART package decoding in the case of VLSI circuit of the USB–UART interface converter.

This demonstrates the ability to completely describe the structure and algorithm of the system with the xtUML diagrams.

4. **Procedure of using xtUML for developing system models.**

System models design in xtUML implies three stages:

- preparatory stage,
- development stage,
- model implementation stage.

The preparatory stage includes the following activities:
• clarification of technical requirements for the system and their synchronization with the use cases,
• splitting the system into domains,
• identification of new and reused domains,
• assessment of key risks of the system project,
• assessment of work amount and its cost.

The development stage includes iterative design, which includes the following stages:

• update of the system domain chart (if necessary),
• creation of a system class diagram,
• creation of a system state diagram,
• adding operations to the classes,
• by means of procedures, indication of operations and action states within the classes,
• by means of procedures, indication of initial conditions and test procedures,
• bridging the domains,
• scenario simulation.

Upon completion of the development phase, the engineer receives an abstract model of the system being developed.

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

As a result of the above procedure, the engineer receives a well-functioning high-level model of the system being developed. It is suitable for further design of digital VLSI circuits at lower levels — the system description can be obtained in low-level languages [3-5, 9]. Figure 9 illustrates the ways of further use of the xtUML modelling.

![Diagram](image)

**Figure 9.** Ways to use xtUML modelling results.

The xtUML model generates two C/C++ codes for two sources of the same system. The first of them is converted to machine code of the pre-configured processor core logic. The second one is converted into a Verilog / VHDL description of the programmable system logic.

Summing up the analysis of the procedure for using the xtUML at the initial stages of the digital hardware components design flow, it should be noted that this procedure fully utilizes the capabilities of the xtUML diagram language, and when using it, the probability of errors at the system design stage is minimized.
5. Conclusion
System level design is the first stage in the digital VLSI circuits design flow. Pertaining to this level, the xtUML diagram-based language provides simulation and verification capabilities for the projects, and also provides the system engineer with the basis for further low-level VLSI design.

Acknowledgements
Some results of the project were obtained with the financial support of the Ministry of science and higher education for the project "Fundamental research of methods of digital transformation of the component base of micro-and nanosystems".

References
[1] Legalov A I, Nepomnyashchij O V and Ryzhenko I V 2018 The Method of Architecturally Independent High-Level Synthesis of VLSI Izvestiya SFedU. Engineering Sciences 8 38-47
[2] Steshenko V B, Rutkevich A V and Gladkova E 2009 Designing of VLSI type "System on a chip". The design route. The synthesis scheme. Part 1 Electronic components 1 14 –21
[3] Demin A A and Vlasov A I 2017 Visual methods of formalization of knowledge in the conditions of the synchronous technologies of system engineering ACM International Conference Proceeding Series p 3166098
[4] Gonoshilov D S and Vlasov A I 2019 Simulation of manufacturing systems using BPMN visual tools Journal of Physics: Conference Series 1353 p 012043
[5] Zhuravleva L V, Vlasov A I and Shakhnov V A 2019 Visual environment of cognitive graphics for end-to-end engineering project-based education Journal of Applied Engineering Science 17 99 -106
[6] Mellor S J and Balcer M J 2001 Executable UML: A Foundation for Model-Driven Architecture (Boston, US/Addison-Wesley Professional) p 161
[7] CP2102/9 Single-Chip USB-to-UART Bridge. Rev. 1.8 1/17. Retrieved from https://www.silabs.com/documents/public/data-sheets/cp2102-9.pdf
[8] Schmuller J 2004 Sams Teach Yourself UML in 24 Hours (Carmel, Indiana, US: Sams Publishing) p 504
[9] Akhter F 2017 Unlocking digital entrepreneurship through technical business process Entrepreneurship and Sustainability 5(1) 36-42