УДК 004.91 DOI: 10.31114/2078-7707-2022-3-18-23

Системный проект ПЛИС с использованием визуальных языков

А.И. Власов, А.А. Расюк, О.Д. Хохлунова

Московский Государственный Технический Университет им. Н. Э. Баумана, г. Москва
rasyuk2a@gmail.com

***Аннотация* — Работа посвящена анализу перспектив использования языка визуальных языков моделирования для синтеза системного проекта ПЛИС. Проанализированы общие подходы использования визуальных языков и выявлены основные направления их применения при проектировании (синтезе) проекта ПЛИС. Показано, что применение визуальных языков позволяет ускорить и упростить процесс проектирования посредством повторного использования уже имеющихся шаблонов архитектуры.**

***Ключевые слова* — системное проектирование, ПЛИС, визуальное моделирование, визуальные языке, UML, маршрут проектирования.**

I. Введение

Существует множество разнообразных методов проектирования цифровых систем. В настоящее время широко распространенной тенденцией является проектирование интегральных схем для конкретных целей на основе программируемых логических интегральных схем (ПЛИС). ПЛИС становятся все более популярными для создания прототипов и проектирования сложных аппаратных систем [1].

Гибкость ПЛИС открыли совершенно новое направление в высокопроизводительных вычислениях. Структура ПЛИС представляет собой “массив логических блоков”, соединенных между собой программируемыми связями. Основными элементами ПЛИС являются конфигурируемые логические блоки. Миллионы логических вентилей в сочетании с возможностью перепрограммирования и реконфигурации дают широкие возможности по разработке [2, 3].

Исходя из вышесказанного можно сделать вывод, что эффективное проектирование цифровых систем на основе ПЛИС требует использования систем автоматизированного проектирования и абстрактного моделирования. Основная цель данной работы – рассмотреть и проанализировать методы использования визуальных языков для визуального описания системного проекта цифровых систем на начальных уровнях разработки.

На начальном этапе проектирования цифровых схем разрабатывается модель системы, включающая поведенческое описание и окружение разрабатываемой системы, позволяющее отразить взаимодействие логической схемы с другими элементами оборудования или объектами измерений. Такой подход эффективен при разработке современных однокристальных систем высокой сложности. И если проводить аналогии с концепциями из единой системы конструкторской документации, то в результате проектирования системы инженер должен получить какую-то структурную схему разрабатываемого изделия. Однако при дальнейшем переходе от системного уровня к уровню, лежащему в основе иерархической декомпозиции, возникает проблема в сложности автоматизированной интерпретации и формализации "прямоугольников" блок-схемы в таком виде, который был бы воспринят системами автоматического проектирования (САПР) следующего, низкоуровневого этапы проектирования. Этот тип в современных маршрутах проектирования цифровой элементной базы может быть поведенческим описанием системы на C, MATLAB, SystemC и т.д. [1]. Для разработки проекта ПЛИС на начальных стадиях необходимо использование интуитивно понятных представлений логических схем [4, 5]. В качестве основы для реализации системного проекта ПЛИС можно использовать унифицированный язык моделирования UML (Unified Modeling Language) [6, 8].

UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. На основании UML-моделей возможна генерация кода, за счет чего можно осуществить переход от модели логической системы на языках высокоуровневого программирования, таких как: C++, C#, Delphi, Fortran, Java, JavaScript, к визуальному представлению.

Унифицированный язык моделирования (UML) содержит несколько графических инструментов, которые можно использовать для иллюстрации, спецификации, визуализации, построения и документирования программных и логических систем. Первая версия языка (1.0) была представлена в 1995 году, а текущая версия - 2.4.2. UML содержит пять основных моделей (вариантов использования, логическая, компонентов, реализация и деятельности),

которые включают несколько видов диаграмм (структурных и поведенческих). Одна из них - диаграмма состояния, которая определяет набор понятий, которые могут быть использованы для моделирования поведения дискретных систем, поскольку они напрямую ссылаются на определение конечных автоматов состояния (FSM (расшифровка)). Графически машина состояний представлена в виде направленного графа с узлами, связанными с состояниями, и ребрами, связанными с переходами между состояниями [5, 9]. Другими словами, это графическое представление дискретного поведения систем с переходами между состояниями.

II. Анализ типового маршрута проектирования цифровой элементной базы

В процессе разработки цифровых логических схем в рамках маршрута проектирования выделяются различные уровни абстракции [10-13]. Так, на рис. 1 показан типовой маршрут проектирования цифровой логической схемы [1]. Из него видно, что, в зависимости от уровня представления, объектом абстракции является система, регистр, вентиль, геометрия библиотечного элемента на кристалле.

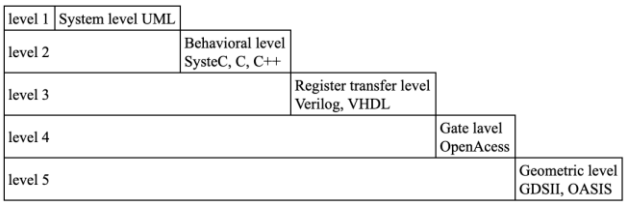

Рис. 1. Типовой маршрут проектирования цифровой элементной базы

Системный уровень — это описание системы на уровне блок-схемы на языке диаграмм. Поведенческий уровень описания проекта состоит из поведенческого описания системы в терминах функций, выражений, алгоритмов. Уровень передачи регистров представляет собой набор арифметических и логических узлов, элементов памяти. Вентильный (логический) уровень описывает проект на уровне логических элементов и триггеров. Самый низкий – геометрический уровень. На нём логические элементы представляются на кремниевом (топологическом) уровне в виде топологических элементов и межсоединений. [7]

Из анализа маршрута проектирования цифровой элементной базы видно, что по мере того, как разработчик цифровой схемы спускается по уровням иерархической декомпозиции в процессе проектирования, с каждым новым этапом ему приходится оперировать менее абстрактными и более конкретными, сложными структурами. Таким образом, проще и выгоднее синтезировать новые и отлаживать текущие решения на системном уровне.

III. Методика использования диаграмм машины состояний для поведенческого выражения системы

Машины состояний могут быть использованы для выражения поведения части системы. Формализм машин состояний, описанный в UML, представляет собой объектно-ориентированный вариант диаграмм состояний Харела. На диаграмме состояние моделируют ситуацию, во время которой выполняется некоторое инвариантное условие. Инвариант может представлять собой статическую ситуацию, например объект ожидает наступления некоторого внешнего события. Однако он также может моделировать динамические условия. Переход — это направленная связь между вершинами-источниками (состояниями) и вершинами-получателями. Он может быть частью составного перехода, который переводит машину состояний из одной конфигурации состояния в другую, представляя собой полную реакцию машины состояний на возникновение события определенного типа.

Диаграммы машин состояний позволяют проектировщику моделировать систему на выбранном иерархическом уровне [9]. Если нет необходимости представлять все детали проектируемой системы, он может выбрать более высокий уровень иерархии и скрыть неважную (на проектируемом уровне) информацию.

Таким образом можно сделать вывод, что диаграммы машин состояний являются одними из наиболее эффективных и важных диаграммами UML для графической спецификации цифровых систем [10]. С учетом параллелизма и иерархии системы проектирования, а также точной семантики, диаграмма машины состояний позволяет разработать полное и однозначное описание поведения цифровой системы.

Пример диаграммы машины состояний приведен на рисунке 2.

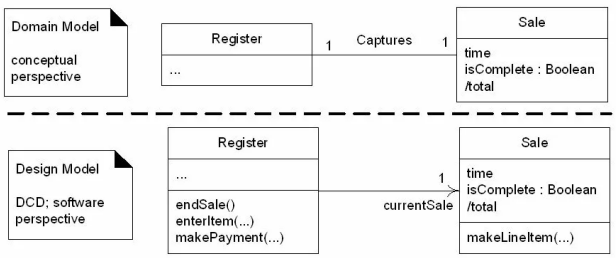

Рис. 2. Диаграмма машины состояний

IV. Методика синтеза проекта ПЛИС с использованием визуальных языков

Чтобы улучшить процесс проектирования и поддержать генерацию кода с мощью инструментов, профиль UML 2.0 для VHDL может быть адаптирован для определения подробной проектной модели реализации языка VHDL.

Профиль определяет структурное представление реализации с помощью диаграмм классов UML и поведение модулей с помощью диаграмм состояний. Эти диаграммы представляют структурную и поведенческую реализацию кода VHDL на этапе проектирования. Набор определенных переходов элементов модели в код VHDL позволяет автоматически генерировать файлы VHDL из модели. [11]

Модель не зависит от конкретной реализации. То есть, одна и та же модель может быть реализована с помощью программного обеспечения, аппаратных средств или смеси того и другого [14]. Различные реализации будут иметь различные характеристики реализации. Выбор реализации зависит от привязки проекта к конкретной аппаратной платформе.

Инструмент адаптируется к различным целевым платформам реализации с помощью модели UML, описывающей ресурсы и ресурсные сервисы, доступные на платформе. Для каждого ресурсного сервиса эта модель описывает характеристики реализации и то, как использовать сервис в реализации [15].

Используя эту информацию, компилятор автоматически сопоставляет проекты UML, с соответствующей платформой реализации. Для каждого реализованного элемента модели дизайна используется подходящая реализация на целевой платформе [16].

Рассматриваемый метод проектирования пытается объединить преимущества графической нотации (читаемость, удобство, интуитивность) с преимуществами языка описания аппаратуры (точность, универсальность).

Можно выделить основные этапы синтеза ПЛИС из UML:

1) На первом этапе проектировщик, используя редактор UML, создает поведенческое описание системы с помощью диаграммы машин состояний UML.

2) Затем диаграмма экспортируется в XML (опция экспорта встроена в каждый профессиональный редактор UML).

3) XML-файлы, соответствующие спецификации XML Metadata Interchange (XMI), являются входными данными для разработанной системы U2V.

4) Система переводит UML-диаграммы в синтезируемое HDL-описание, используя темпоральную модель HCFSM (каждый модуль сохраняется в отдельном файле).

5) Затем, используя внешние инструменты, система может смоделировать, и, наконец, можно выполнить синтез и реализацию.

6) Результатом последнего этапа является Netlist и соответствующий битовый поток, с помощью которых можно программировать реальное устройство ПЛИС.

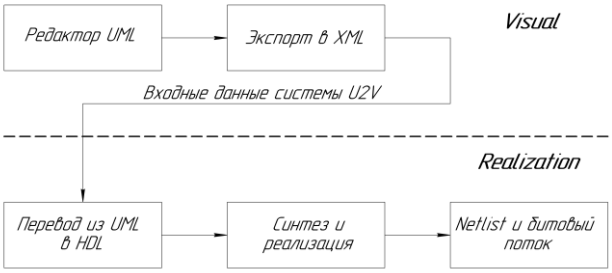

Рис. 3. Алгоритм методики синтеза проекта плис с использованием визуальных языков

В результате применения вышеописанной методики инженер получает отлаженную высокоуровневую модель разрабатываемой системы, пригодную для дальнейшего проектирования цифровой элементной базы на более низких уровнях – получения описания системы на языках низкого уровня.

Подводя итоги анализа методологии использования визуальных языков на начальных этапах маршрута проектирования цифровой элементной базы, можно отметить, что данная методика полностью использует возможности языка UML, и при его использовании вероятность ошибок на этапе проектирования системы сводится к минимуму.

V. Пример построения модели системного проекта ПЛИС с использованием языка UML

В качестве примера мы разработаем диаграммы языка HTML для статической оперативной памяти СБИС (англ. Static Random Access Memory, SRAM). Конечной целью создания модели системы на языке диаграмм UML является синтез кода, который может быть использован на более низких уровнях проектирования системы.

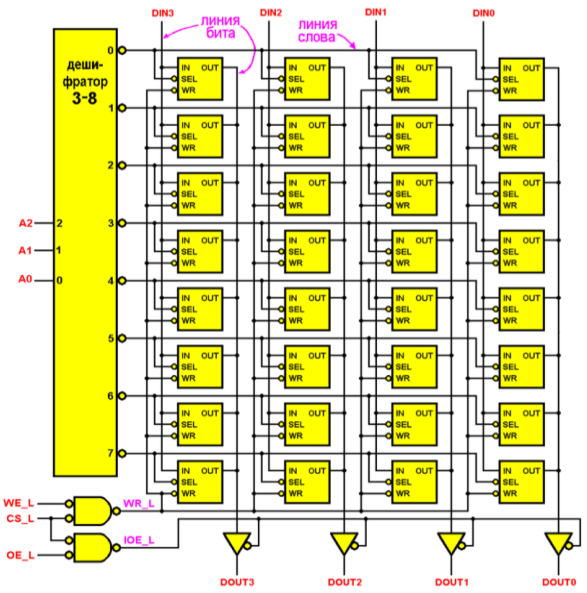

Рис. 4. Структурная схема СБИС статической памяти

Начать работу следует с анализа структурной схемы разрабатываемой системы. Структурная схема СБИС статической памяти представлена на рис. 4.

Чтобы понять, какие функции системы доступны каждому из ее пользователей, мы рассмотрим модель ячейки памяти. Функциональная модель ячейки СБИС статической памяти показана на рис. 5.

Из функциональной модели видно, что СБИС со статической памятью имеет два варианта использования: процесс, когда некоторая внешняя система записывает информацию в память СБИС, и процесс, когда некоторая система считывает информацию из СБИС.

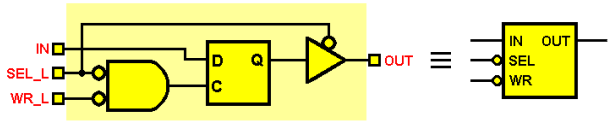

Рис. 5. Функциональная модель ячейки СБИС статической памяти

В соответствии со структурной схемой системы, показанной на рис. 7, мы разделим систему на домены: каждый блок в структурной схеме должен иметь свой собственный домен. Схема доменов модели статической памяти СБИС показана на рис. 6.

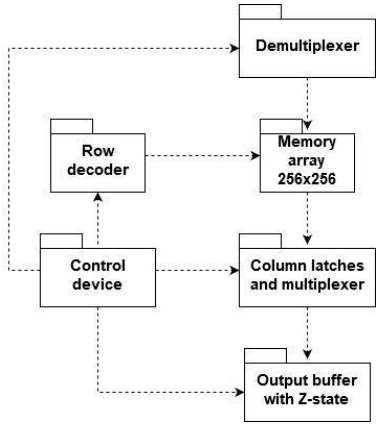

Рис. 6. Диаграмма доменов модели СБИС статической памяти

Диаграмма доменов декомпозирует систему – она разбивает исходную систему на несколько элементарных подсистем.

Создание диаграммы предметной области завершает первый этап разработки модели системы – подготовительный этап. Следующий этап — это этап разработки. Его ключевыми шагами являются:

1) создание диаграммы классов системы,
2) создание диаграммы состояний системы,
3) добавление операций в классы,
4) указание операций и состояний действий в классах с помощью процедур,
5) указание начальных условий и тестовых методик с помощью процедур,
6) симуляция полученного сценария.

Набор атрибутов и методов для диаграммы классов состоит из функциональности, которая требуется от статической памяти, разработанной СБИС. В этом случае атрибутами классов являются частота информационных сигналов, размер шины данных. Методы представляют собой такие функции, как "введите адрес строки", "введите адрес столбца", "разрешить запись данных", "разрешить чтение данных". Диаграмма классов модели статической памяти СБИС показана на рис. 7.

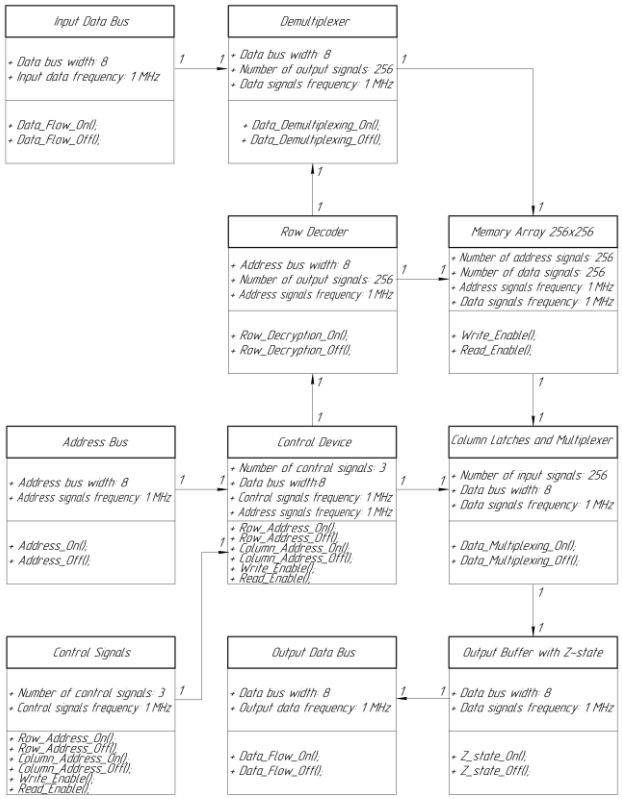

Рис. 7. Диаграмма классов модели СБИС статической памяти

Из диаграммы классов стало ясно, с какими сигналами работает каждый класс и какие функции выполняет каждый из них.

Набор состояний и переходов для диаграммы состояний составляется на основе режимов работы статической памяти BLUE. Режимы работы статической памяти BLUE можно отслеживать по временным диаграммам записи и чтения. Временные диаграммы записи и чтения статической памяти СБИС показаны на рис. 8 и 9.

Состояниями статической памяти СБИС являются "Ожидание команд", "Ввод адреса строки", "Ввод адреса столбца", "Запись в память", "Чтение из памяти". Переходами СБИС статической памяти являются "Активация СБИС", "Поступление сигнала разрешения на чтение", "Поступление сигнала

разрешения на запись", "Начало чтения", "Начало записи", "Конец чтения", "Конец записи". Диаграмма состояний модели статической памяти СБИС показана на рис. 10.

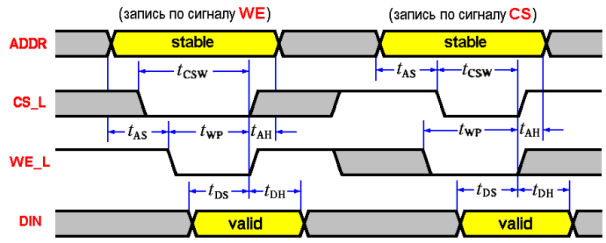

Рис. 8. Временная диаграмма записи данных в СБИС статической памяти

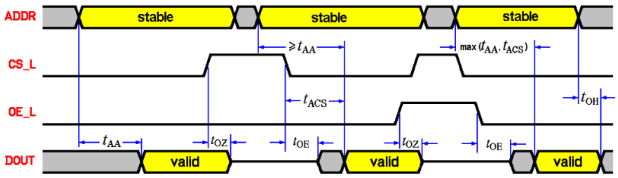

Рис. 9. Временная диаграмма чтения данных из СБИС статической памяти

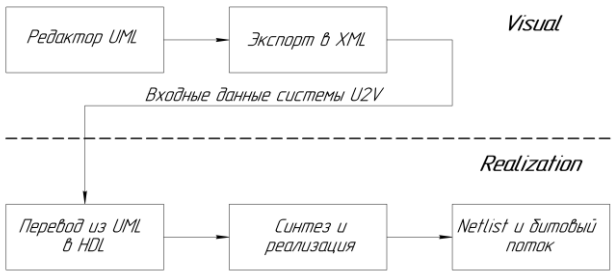

Рис. 10. Диаграмма состояний модели СБИС статической памяти

Диаграмма состояний модели СБИС обеспечивает основу для синтеза конечного автомата системы.

Диаграммы доменов, классов и состояний предоставляют полную информацию о разрабатываемой системе. Далее, после описания диаграмм в Mentor Graphics BridgePoint CAD, вы можете моделировать и отлаживать систему, а также синтезировать ее структуру на языке описания оборудования. Этот шаг позволяет разработчику системы, используя методологию, описанную в разделе IV, получить отправную точку для проектирования системы низкого уровня.

VI. Заключение

Рассмотрен метод графической спецификации цифровых систем, в частности ПЛИС. Метод основан на модели машин состояний UML и генерируется поведенческое, синтезируемое и модульное описание системы на языке описания VHDL. Данные методы применимы к общим подходам генерации описания оборудования. Можно либо создавать системы, в которых все поведение отображается непосредственно на выделенное оборудование, либо создавать системы, содержащие общие наборы определенных или даже общих аппаратных компонентов с логикой управления и динамического назначения этих компонентов для различных операций. Язык диаграмм UML позволяет упростить разработку (синтез) ПЛИС и обеспечивает наглядность при проектировании.

Поддержка

Отдельные результаты получены при финансовой поддержке Министерства науки и высшего образования РФ по проекту №0705-2020-0041 «Фундаментальные исследования методов цифровой трансформации компонентной базы микро- и наносистем»

Литература

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

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

[3] Котельницкий А.В., Власов А.И. Применение поведенческих моделей при проектировании систем на кристалле // Инженерный вестник. 2012. № 9. С. 10.

[4] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Visual methods of high-level system design for digital hardware components // JOP Conference Series: Metrological Support of Innovative Technologies. 2020. N. 42024.

[5] Хальзев С.Е., Власов А.И., Шахнов В.А. Применение визуальных средств для системного моделирования цифровых интегральных схем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2020. Выпуск 1. С. 8-14. doi:10.31114/2078-7707-2020-1-8-14

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

[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] Grzegorz Bazydlo, Marian Adamski, Marek Wegrzyn, Alfredo Rosado Munoz. From UML Specification into FPGA Implementation // Advances in electrical and electronic engineering. 2014. С. 452-457.

[9] G. Bazydlo, M. Adamski, M. Węgrzyn, A. Rosado Munoz, From UML State Machine Diagram into FPGA Implementation // 12th IFAC Conference on Programmable Devices and Embedded Systems The International Federation of Automatic Control September 25-27, 2013. Velke Karlovice, Czech Republic. C. 2-3.

[10] Tim Schattkowsky, Achim Rettberg, UML for FPGA Synthesis // Paderborn University, C-LAB D-33102 Paderborn, Germany. C. 15.

[11] Tonfat, Jorge, A UML Profile For VHDL FPGA Designs // SEFUW: SpacE FPGA Users Workshop, 5th Edition. C. 3.

[12] 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

[13] Иванов А.М., Власов А.И. Верификация программных моделей коммуникационных сетей // Наука и

образование: научное издание МГТУ им. Н.Э. Баумана. 2012. № 10. С. 24.
[14] Власов А.И. Пространственная модель оценки эволюции методов визуального проектирования сложных систем // Датчики и системы. 2013. №9 (172). С. 10-28.
[15] Власов А.И. Концепция визуального анализа сложных систем в условиях синхронных технологий проектирования // Датчики и системы. 2016. № 8-9(206). С. 19-25.
[16] Vlasov A.I., Demin A.A. Visual methods of formalization of knowledge in the conditions of the synchronous technologies of system engineering // ACM International Conference Proceeding Series. CEE-SECR. 2017. N.3166098.

FPGA System Design Using Visual Languages

A.I. Vlasov, A.A. Rasyuk, O.D. Khokhlunova

N.E. Bauman Moscow State Technical University, Moscow

rasyuk2a@gmail.com

***Abstract* — The paper is devoted to the analysis of the prospects of using visual modeling languages for the synthesis of FPGA system design. The general approaches to the use of visual languages are analyzed and the main directions of their application in the design (synthesis) of FPGA project are revealed. It is shown that the use of visual languages can accelerate and simplify the design process by reusing existing architecture templates.**

***Keywords* — system design, FPGA, visual modeling, visual language, UML, design route.**

REFERENCES

[1] Steshenko V.B., Rutkevich A.V., Gladkova E. et al. Design of VLSIs of "System-on-Chip" type. Design route. Circuit synthesis. Part 1 // Electronic components. - 2009. - №1. - P. 14 -21.
[2] Rabovoluk A. Review of Mentor Graphics' FPGA Advantage design route // Components and Technologies. 2005. №5. P. 98-101.
[3] Kotelnitskii A.V., Vlasov A.I. Application of behavioral models in designing systems on a crystal // Engineering Herald. 2012. № 9. P. 10.
[4] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Visual methods of high-level system design for digital hardware components // JOP Conference Series: Metrological Support of Innovative Technologies. 2020. N. 42024.
[5] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Application of visual tools for system modeling of digital integrated circuits // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2020. Issue 1. P. 8-14. doi:10.31114/2078-7707-2020-1-8-14
[6] Stephen J. Mellor, Marc J. Balcer, Executable UML: A Foundation for Model-Driven Architecture // Addison-Wesley Professional. 2001. 161 p.
[7] Vlasov A.I., Karpunin A.A., Ganev Yu.M. System approach to design for cascade and iterative life cycle model // Proceedings of the International Symposium "Reliability and Quality". 2015. T. 1. P.96-100.
[8] Grzegorz Bazydlo, Marian Adamski, Marek Wegrzyn,Alfredo Rosado Munoz. From UML Specification into FPGA Implementation // Advances in electrical and electronic engineering. 2014. P. 452-457.
[9] G. Bazydlo, M. Adamski, M. Węgrzyn, A. Rosado Munoz, From UML State Machine Diagram into FPGA Implementation // 12th IFAC Conference on Programmable Devices and Embedded Systems The International Federation of Automatic Control September 25-27, 2013. Velke Karlovice, Czech Republic. P. 2-3.
[10] Tim Schattkowsky, Achim Rettberg, UML for FPGA Synthesis // Paderborn University, C-LAB D-33102 Paderborn, Germany. P. 15.
[11] Tonfat, Jorge, A UML Profile For VHDL FPGA Designs // SEFUW: SpacE FPGA Users Workshop, 5th Edition. P. 3.
[12] Butch G., Rambaud D., Jacobson I. UML Language. User's Guide. 2nd ed.: Translated from English by Mukhin N. - M.: DMK Press. - 496 p.: ill. ISBN 5-94074-334-X.
[13] Butch G., Rambaud D., Jacobson I. UML Language. User's Guide. 2nd ed.: Translated from English by Mukhin N. - M.: DMK Press. - 496 p.: ill. ISBN 5-94074-334-X.
[14] Vlasov A.I. Spatial model for evaluating the evolution of methods of visual design of complex systems // Sensors and Systems. 2013. №9 (172). P. 10-28.
[15] Vlasov A.I. The concept of visual analysis of complex systems under synchronous design technologies // Sensors and Systems. 2016. № 8-9(206). P. 19-25.
[16] Vlasov A.I., Demin A.A. Visual methods of formalization of knowledge in the conditions of the synchronous technologies of system engineering // ACM International Conference Proceeding Series. CEE-SECR. 2017. N.3166098.