FRI-2G.303-1-CCT1-12

LEARNING VHDL USING THE CONCEPT OF VISUAL PROGRAMMING [12]

Assoc. Prof. Aneliya Ivanova, PhD
Department of Computer Systems and Technologies,
University of Ruse "Angel Kanchev"
Phone: 082-888 827
E-mail: aivanova@uni-ruse.bg

Venelin Mandov, MSc.
Department of Computing,
University of Ruse, Bulgaria
E-mail: venelinmandov98@gmail.com

Principal Assistant Nikolay Kostadinov, PhD
Department of Computing,
University of Ruse, Bulgaria
E-mail: nkostadinov@uni-ruse.bg

Abstract: *When the students studying programmable logic design have to learn a hardware description language (HDL) in order to implement their projects, they usually meet several challenges. The first one of them is due to the development environments for HDLs which are usually designed for highly trained professionals and could be complicated as an introductory tool for students learning HDLs. The next challenge is for the students to start thinking in terms of hardware when writing HDL code, and have in mind that each HDL structure models a digital device. The further challenge is for the students to get used to think in terms of signals, ports and processes running parallel to each other, rather than of variables, methods and functions. HDLs are compiled to a RTL structure, which requires a logically justified construction of the code, describing the behaviour of the modelled device. In this paper is presented the process of design and development of a training environment that uses the visual programing concept to help the students to become familiar with the structure and hierarchy of the VHDL models. The students construct VHDL models by selecting and assembling together simple visual objects, while the environment is providing guidance in real time and is preventing wrong matches of VHDL operators and signals. The visual programming approach has been chosen because the digital students as a whole have visual-kinaesthetic learning style and most of them are familiar with visual programming tools, such as Scratch and Kodu from IT classes at school. The environment prompts the student to select a template from a library, thus avoiding inconsistent structures and combinations of devices in the project. When a template is selected, the "Visual Components" panel of the environment displays only the appropriate visual components, which correspond to the template. After the visual structure is assembled correctly, the student is offered to start its' conversion to VHDL code. The output VHDL code is saved as a .vhd file and can be further open with any IDE supporting VHDL.*

Keywords: *Hardware Description Language, VHDL, Interactive Learning Environment, Visual Programming*
JEL Codes: *I21, I23.*

ВЪВЕДЕНИЕ

В контекста на глобалните промени, с които се сблъска човечеството през последните две години, динамичния напредък на дигиталните технологии и мащабните инициативи за дигитализация на почти всички сфери на човешката дейност, онлайн обучението започва да придобива все по-важна роля във висшето образование и необходимостта от разработване на образователни инструменти, които ще помогнат на студентите да учат по-добре по всяко време и от всяко място, вече не може да бъде пренебрегната. Онлайн преподаването на езици за програмиране е само по себе си предизвикателство, а когато предмет на обучението е

[12] Докладът е представен на сесия на секция 3.2 на 28 октомври 2022 с оригинално заглавие на български език: ИЗУЧАВАНЕ НА VHDL С ПОМОЩТА НА КОНЦЕПЦИЯТА ЗА ВИЗУАЛНО ПРОГРАМИРАНЕ

проектирането с програмируема логика чрез езици за описание на хардуер (HDL), това предизвикателство става още по-сериозно.

Средите за проектиране с помощта на HDL поради своята специфика могат да се окажат доста сложни за усвояване за начинаещите в тази област студенти, така че последните се нуждаят от повече указания и пояснения от страна на преподавателя. Предоставянето на обратна връзка под формата на корекции и оценка на HDL проектите на студентите в режим на онлайн обучение също е сложна задача.

В (Kumar, A., R., Panicker, C., Kassim, A., 2013) се представя среда с отворен код за синтез и симулация на VHDL (Very High Speed Integrated Circuit Hardware Description Language) модели, осигуряваща автоматизирана проверка на студентски проекти и обратна връзка за функционалната коректност на проектите. В друга разработка, симулационен модел на мрежова карта, базиран на VHDL, се използва като педагогически инструмент, който помага на студентите да разберат по-добре принципите на компютърните архитектури (Godofredo, R. G., Tchernykh, A., Yu, A., Drozdov, Garichev, S. N., Nesmachnow, S., Torres-Martinez, M., 2019). Един подход за електронно оценяване на VHDL код, реализиран като модул, интегриран в платформата Moodle, е представен в (Jelemenska, K., Cicak, P., Gazik, M. 2016). Внедреният модул проверява дали кодът на студента може да бъде успешно компилиран и дали симулацията на този код е идентична с референтната симулация.

ИЗЛОЖЕНИЕ

Предпоставки

Дисциплината "*Технология на проектирането*" от бакалавърската учебна програма на специалност "*Компютърни системи и технологии*" запознава студентите с концепциите на езика VHDL и принципите на проектирането с програмируема логика.

От гледна точка на студентите първото предизвикателство е да започнат да мислят на ниво хардуер, когато пишат VHDL код и да асимилират идеята, че всяка VHDL структура моделира цифрово устройство. Второто предизвикателство е да свикнат с идеята, че оперират със сигнали, входно-изходни портове и процеси, протичащи паралелно един на друг, а не с променливи и функции, както в езиците за програмиране. VHDL изисква стриктно деклариране на типовете данни и сигнали в даден проект, което налага да се познават добре особеностите на езика – къде се декларират вътрешни сигнали, променливи, типове данни и др. VHDL кодът се компилира в RTL (Register-Transfer Level) структура, а това изисква не само отлично познаване на принципа на работа на моделираното устройство, но и писане на логически издържан код, описващ поведението му. И накрая, студентите трябва добре да познават структурата и йерархията на различните видове VHDL модели, за да могат да създават проекти с различна сложност.

В този доклад са представени проектирането и ра заработката на обучаваща среда, която използва концепцията за визуално програмиране, за да помогне на студентите да създават VHDL модели на различни цифрови устройства. Конструирането на VHDL модели се осъществява чрез съчетаване и подреждане на прости визуални обекти, като студентите получават своевременна обратна връзка с упътване и съобщения за грешки, за да се избегне неправилното комбиниране на VHDL оператори и сигнали.

Подходът за визуално програмиране е обсъждан в редица изследвания като добра техника за подпомагане изучаването на принципите на езиците за програмиране (Seraj, M., Autexier, S., Janssen, J., 2018), (Pinto-Llorente, A.M., Casillas-Martín, S., Cabezas-González, M., et al., 2018), (Baldwin, L. P., Kuljis, J., 2000), (Carlisle, M. C., Wilson, T. A., Humphries, J. W., Hadfield, S. M., 2005), (Xie, C. , Qi, H., Ma, L., Zhao, J., 2019), (Al-Tahat, K., 2019). В настоящата разработка този подход е избран поради обстоятелството, че дигиталните студенти като цяло имат визуално-кинетичен стил на учене, а инструментите за визуално програмиране като Scratch и Kodu се използват в училищното образование и са доказано ангажиращи учебни среди за въведение в принципите на програмирането (Rose, S. P., Habgood, J. M. P., Jay, T., 2017).

Много от студентите, изучаващи „*Технология на проектирането*“ са запознати с тези среди от часовете по информационни технологии в училище.

При практическите занятия по дисциплината студентите използват два подхода за моделиране: чрез структурно и поведенческо описание, като моделите на сложните цифрови устройства могат да се реализират чрез йерархия от .vhd файлове, представена в схематичен вид или описана във файл със структурна архитектура или чрез единичен .vhd файл с поведенческа архитектура, в който всеки модул от сложното устройство е моделиран като отделен процес в рамките на архитектурата. Тъй като при втория подход се проектът се разполага в един .vhd файл, е целесъобразно да бъде избран именно той за създаване на VHDL модела, чийто код трябва да бъде генериран от обучаващата среда.

Изисквания към средата

Функционалността на средата трябва да подпомага изучаването на структурата на VHDL модела, основните оператори и синтаксиса на езика и начина на моделиране на цифровите устройства. За целта е необходимо:

◇ средата да поддържа библиотека от визуални компоненти, съответстващи на отделните части на VHDL модела, служебни думи, основните оператори, както и на примерни цифрови устройства, които да се използват в разработваните проекти;

◇ библиотеката от визуални компоненти, която ще се ползва в средата, да бъде съобразена със съдържанието на дисциплината и спецификата на проектите, разработвани от студентите като практически упражнения и курсови задачи;

◇ средата да предлага възможност на студента да избере шаблон на проект от библиотека с предварително подготвени такива шаблони, като по този начин да се избегне сглобяването на неподходящи структури от оператори или комбинации от устройства в проекта;

◇ интерфейсът на приложението да бъде организиран в няколко работни зони – за работа с библиотеката от визуални компоненти и готови шаблони, зона за разполагане на избрания шаблон, работна площ за конструиране на VHDL модела и зона за извеждане на генерирания код;

◇ средата да позволява избор на визуален обект от съответна категория, разполагане на обекта в работната площ и неговото преместване, редактиране и премахване по всяко време в процеса на конструиране на VHDL модела;

◇ средата да позволява проверка на валидността на конструкцията по всяко време в процеса на изграждането ѝ и да извежда съобщения за грешки и подсказки за тяхното отстраняване;

◇ генерирането на VHDL кода да бъде възможно само след успешна валидация на завършената конструкция;

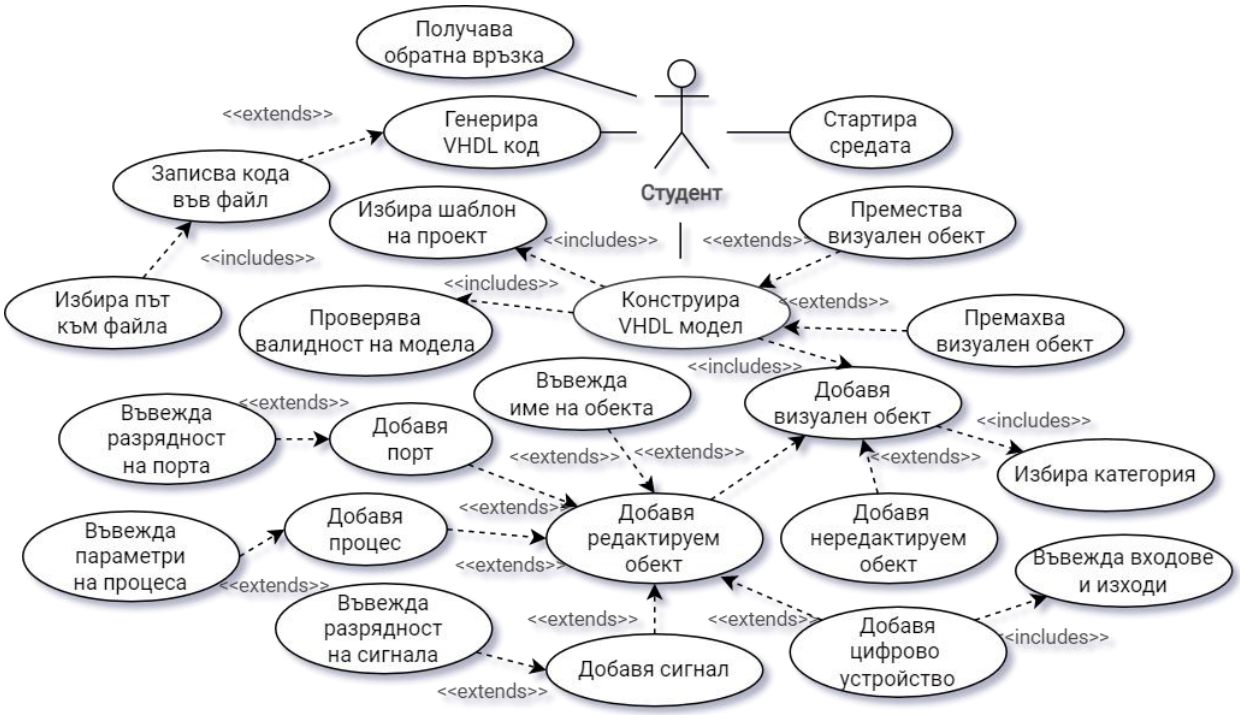
◇ след генерирането на VHDL кода средата да позволява записването му в изходен файл във .vhd формат, който да може да бъде асоцииран с нов или съществуващ VHDL проект;

◇ за да бъде горното възможно, при генерирането на VHDL кода автоматично да се декларират и най-често използваните стандартизирани библиотеки, които се ползват по време на практически упражнения.

Проектиране

Функционалният модел на средата е представен на фиг. 1 и 2 под формата на диаграми на случаите на употреба и на дейностите. Проектите, които студентите разработват, моделират устройства, включващи структури от последователностни и комбинационни схеми като регистри, броячи, дешифратори, приоритетни шифратори, кодови преобразователи, мултиплексори, компаратори и др., както и някои по-сложни устройства като крайни автомати с памет, блокове памет, АЛУ и т.н. В тази пилотна версия на средата е избрана концепцията студентът да избира проектен шаблон от предварително подготвена библиотека, с което се постига по-лесна верификация на конструирания VHDL модел, както и по-прецизна обратна връзка и по-конкретни подсказки.

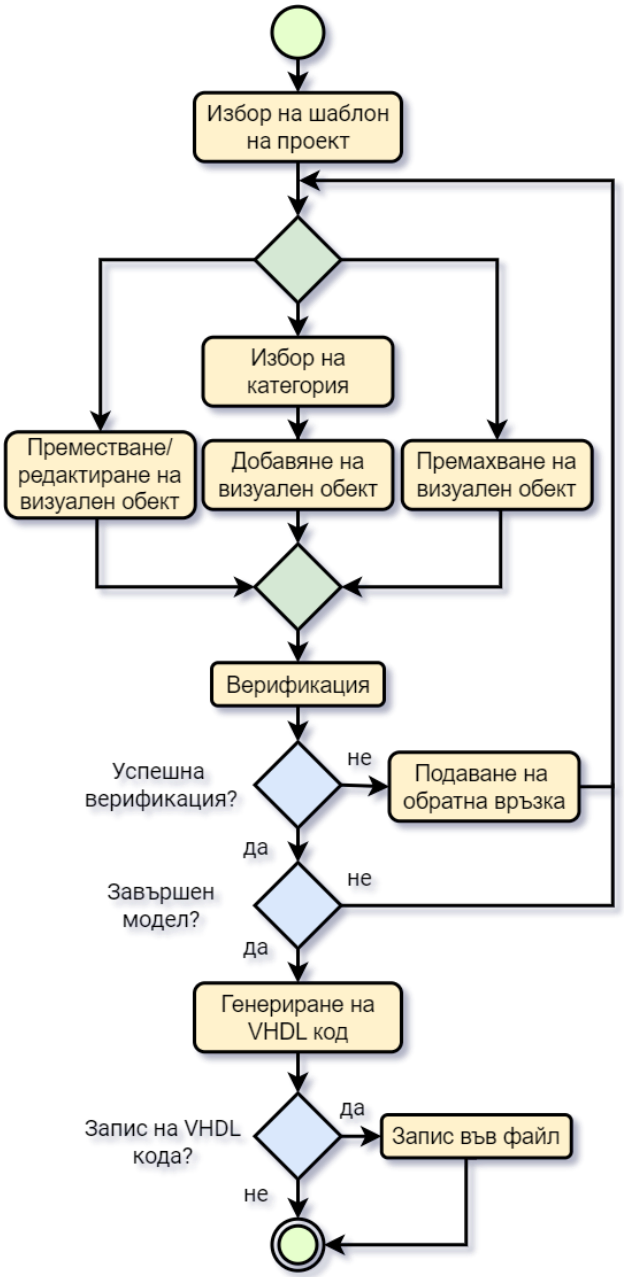
След като е избран проектен шаблон, панелът „*Визуални компоненти*“ показва само подходящите визуални компоненти, които съответстват на шаблона. Студентът, след като избере една от предложените категории (*Декларативна част*, *Компоненти на VHDL модела*, *Конкурентни оператори*, *Последователни оператори*, *Цифрови логически устройства*) може да избира, разполага, премества и премахва визуални обекти от конструкцията. Според възможността да бъдат редактирани или не, визуалните обекти и блокове се класифицират като редактируеми и нередактируеми, което означава, че студентите могат да въвеждат имена и параметри в декларациите на входни и изходни портове, вътрешни сигнали, процеси и др., или да въвеждат имена на входовете и изходите на визуален блок, представляващ конкретна логическа схема.



Фиг. 1. Функционален модел – случаи на употреба

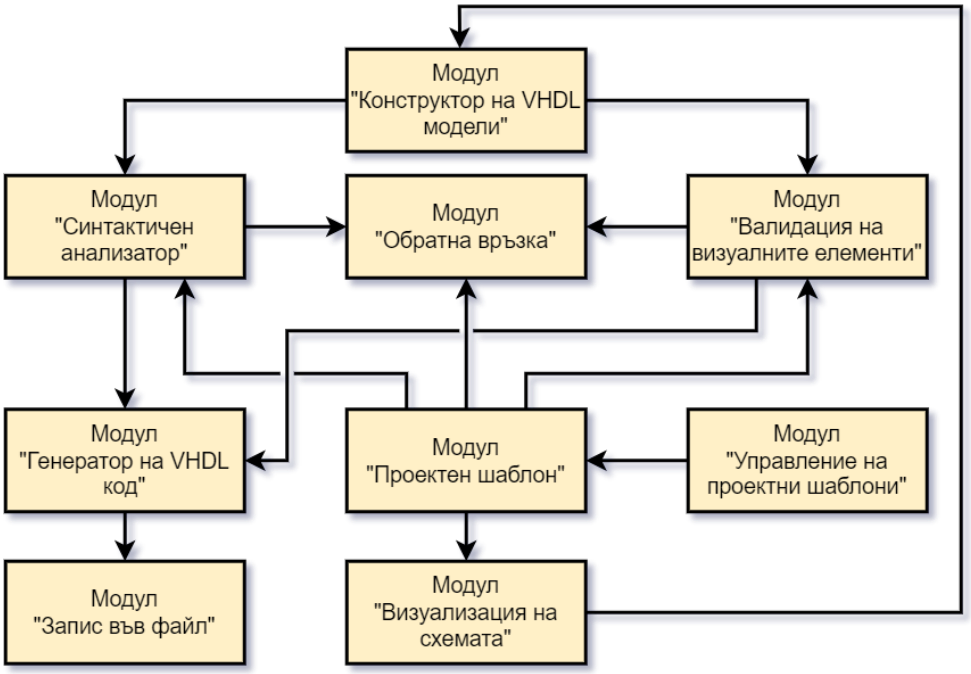
Както се вижда от фиг. 2, студентът има възможност по всяко време да стартира валидация на конструкцията, но само след като визуалната конструкция е завършена и сглобена правилно, може да се стартира преобразуването във VHDL код. След извеждане на генерирания VHDL код, той може да бъде записан в изходен файл.

На фиг. 3 е представена архитектурата на разработената система. Два модула следят конструкцията от визуални обекти, която студентът сглобява в работната площ.



Фиг. 2. Функционален модел – диаграма на дейностите

Функциите на модула „*Синтактичен анализатор*“ наподобяват лексичния и синтактичния анализ в един класически компилатор и включват поддържане на таблица с откритите и класифицирани визуални обекти с техните атрибути, следи подредбата им по предварително зададената синтактична структура на избрания шаблон и генерира съобщения за грешки, а модулът „*Валидация на визуалните елементи*“ извършва семантичен анализ и проверява съвместимостта на атрибутите на входните и изходните портове и вътрешните сигнали, както и входовете и изходите на визуалните блокове и също подготвя съобщения за грешки. Модулът „*Обрата връзка*“ извежда съобщенията за грешки, получени от горните два модула. Модулът за генериране на VHDL код, в зависимост от класа и атрибута на всеки визуален обект от конструкцията, поставя в негово съответствие VHDL код, предварително записан като символен низ в масив.



Фиг. 3. Основни модули на системата

Реализация

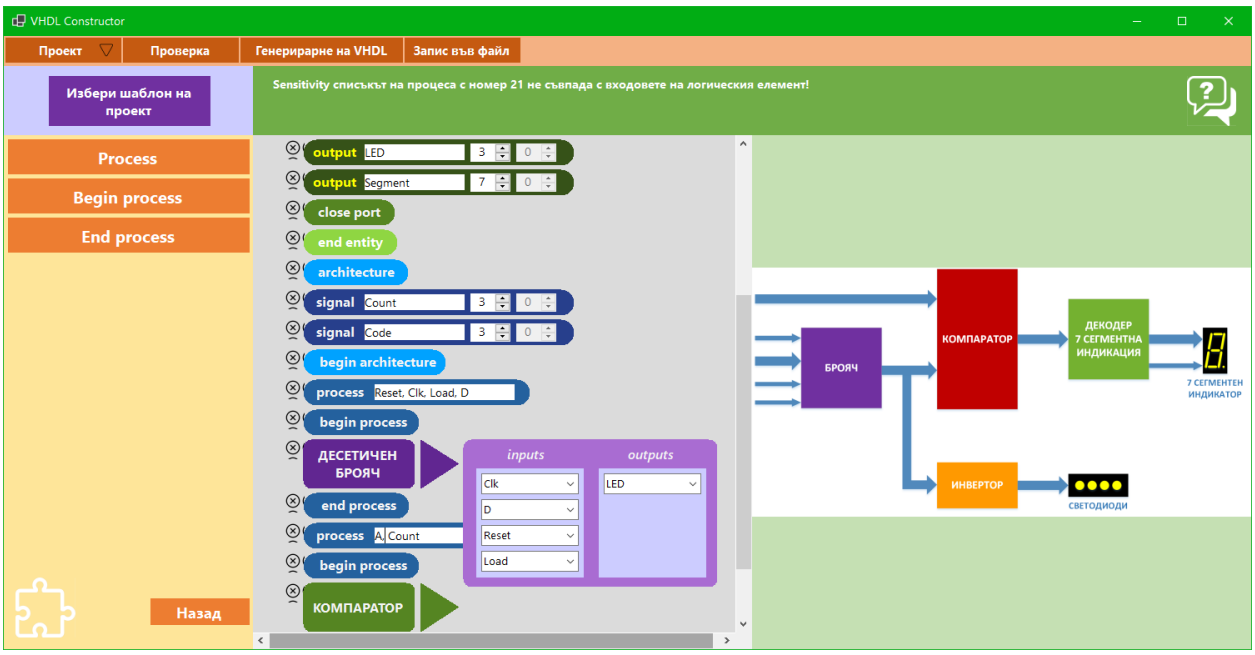
Обучаващата среда е реализирана като настолно приложение с помощта на езика за програмиране C#, платформата .NET и графичната интерфейсна библиотека Windows Forms. На фиг. 4 е представен екран на приложението в процес на конструиране на VHDL модела. Потребителският интерфейс на средата наподобява структурата на интерфейса на типична IDE (интегрирана среда за разработка), с тази разлика, че панелът за обратна връзка се намира в горната част на прозореца, тъй като последната играе важна роля във всяка образователна среда.

Интерфейсът е организиран в 6 области, като 4 от тях са зони на активно взаимодействие, като лентата „*Меню*“, панелът за избор на проектен шаблон, панелът за избор на визуални компоненти и панелът за конструиране на VHDL модела.В лентата с менюта има едно падащо меню, предоставящо всички опции, отнасящи се до проекта – нов проект, запазване, запазване под ново име и затваряне, а останалите елементи от менюто са бутони за валидация на проекта, за генериране на VHDL код и за записване на генерирания код във .vhd файл.

Организацията на панела за избор на визуални компоненти съответства на дефинираните категории и подкатегории на визуалните обекти и когато е избран конкретен шаблон, категориите, които не са необходими за проекта, се скриват за потребителя. Панелът за избор на визуални компоненти и работната площ за конструиране на VHDL модела са най-често използваните области на интерфейса. Студентът избира визуални обекти и блокове и ги поставя в зоната за конструиране чрез операция drag-drop, докато завърши структурата на VHDL модела.

Първата зона на пасивно взаимодействие има две функции в хода на работа по проекта. Първоначално в нея се показва схемата на избрания проектен шаблон, както се вижда от фиг. 4, а след генерирането на VHDL кода, последният се извежда в същата зона. Втората пасивна област е панелът за обратна връзка, където студентът получава насоки в реално време под формата на съвети за най-добро комбиниране на визуални компоненти или предупреждения и съобщения за грешки. Единственото активно действие в този панел е изборът на опцията „Помощ“ за по-подробно ръководство с инструкции.

Както може да се види от фиг. 4, визуалните обекти заместват декларации, оператори, ключови думи, а визуалните блокове заместват логически устройства.



Фиг. 4. Екран на приложението в процес на конструиране на VHDL модела

ЗАКЛЮЧЕНИЕ

Подходът за визуално програмиране и стратегията за изграждане на модел на сложно цифрово устройство чрез преместване и свързване на визуални обекти ще помогнат на студентите с визуално-кинетичен стил на учене да усвоят концепцията и структурата на VHDL модела.

Ръководството, предоставено от средата като предупреждения и препоръки при откриване на опити за свързване на несъвпадащи обекти, ще научи студентите как да изграждат правилно VHDL структура и ще формира подходящи умения за развойна дейност.

Опцията за избор на проектен шаблон ще помогне на начинаещите студенти по-лесно да разберат структурата на проектите, които разработват и ще улесни осигуряването на по-добро ръководство от страна на средата по време на конструкция на VHDL модела.

Обучаващата среда ще бъде от полза най-вече в процес на самоподготовка на студентите, и особено при разработването на курсовата задача по дисциплината, но може да се използва и като нагледно средство в процеса на преподаване или като помощен инструмент при провеждане на онлайн упражнения.

Като насока за бъдещо развитие и усъвършенстване на средата се предвижда възможността VHDL моделът да бъде конструиран и без предварителен избор на проектен шаблон.

REFERENCES

Kumar, A., R., Panicker, C., Kassim, A. (2013). *Enhancing VHDL learning through a light-weight integrated environment for development and automated checking*. Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering, pp. 570-575. doi: 10.1109/TALE.2013.6654502.

Godofredo, R. G., Tchernykh, A., Yu, A., Drozdov, Garichev, S. N., Nesmachnow, S., Torres-Martinez, M. (2019). *Visualization of VHDL-based simulations as a pedagogical tool for supporting computer science education.* Journal of Computational Science, Volume **36**, doi: 10.1016/j.jocs.2017.04.004.

Jelemenska, K., Cicak, P., Gazik, M. (2016). *VHDL models e-assessment in Moodle environment.* International Conference on Emerging eLearning Technologies and Applications (ICETA), Vysoke Tatry, pp. 141-146, doi: 10.1109/ICETA.2016.7802048.

Seraj, M., Autexier, S., Janssen, J. (2018). *BEESM, a block-based educational programming tool for end users*. In Proceedings of the 10th Nordic Conference on Human-Computer Interaction (NordiCHI '18). Association for Computing Machinery, New York, NY, USA, 886–891. doi: 10.1145/3240167.3240239

Pinto-Llorente, A.M., Casillas-Martín, S., Cabezas-González, M., et al. (2018). *Building, coding and programming 3D models via a visual programming environment*. Qual Quant 52, 2455–2468. doi: 10.1007/s11135-017-0509-4

Baldwin, L. P., Kuljis, J. (2000). *Visualisation techniques for learning and teaching programming.* ITI 2000. Proceedings of the 22nd International Conference on Information Technology Interfaces (Cat. No.00EX411), Pula, Croatia, pp. 83-90, doi: 10.2498/cit.2000.04.03.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., Hadfield, S. M. (2005). *RAPTOR: a visual programming environment for teaching algorithmic problem solving.* SIGCSE Bull. **37, 1**, pp.176–180. doi: 10.1145/1047124.1047411.

Xie, C. , Qi, H., Ma, L., Zhao, J. (2019) *DeepVisual: A Visual Programming Tool for Deep Learning* Systems 27th International Conference on Program Comprehension (Montreal: Canada/IEEE/ACM) pp. 130-134, doi: 10.1109/ICPC.2019.00028.

Al-Tahat, K. (2019) *The Impact of a 3D Visual Programming Tool on Students' Performance and Attitude in Computer Programming: A Case Study* in Jordan Journal of Cases on Information Technology (JCIT) **21(1)**, doi: 10.4018/JCIT.2019010104.

Rose, S. P., Habgood, J. M. P., Jay, T. (2017) *An Exploration of the Role of Visual Programming Tools in the Development of Young Children's Computational Thinking*. The Electronic Journal of e-Learning, Volume **15,** Issue **4** pp. 297-309.

ACKNOWLEDGEMENT

This paper is supported by project 2022–EEA–01"Analysis of big data processing algorithms and their application in multiple subject domains", funded by the Research Fund of the "Angel Kanchev" University of Ruse.