Corso Pinzón et al. *Research and Practice in Technology Enhanced Learning* (2024) 19:15
https://doi.org/10.58459/rptel.2024.19015

Research and Practice in Technology Enhanced Learning

RESEARCH Free and Open Access

Automated grading software tool with feedback process to support learning of hardware description languages

Andrés Francisco Corso Pinzón, Jhon J. Ramírez-Echeverry * and Felipe Restrepo-Calle

*Correspondence:
jjramireze@unal.edu.co
Electrical and Electronics Department,
Universidad Nacional de Colombia,
Carrera 45 # 45-03 Building 453 Office 210, Bogotá D.C, 111321, Colombia
Full list of author information is available at the end of the article

Abstract

Hardware Description Languages (HDL) have gained popularity in the field of digital electronics design, driven by the increasing complexity of modern electronic circuits. Consequently, supporting students in their learning of these languages is crucial. This work aims to address this need by developing an automated assessment software tool with feedback process to support the learning of HDL and making an educational intervention to support the learning process of students. The tool's features were selected based on similar developments, and a prototype was designed and implemented. Additionally, an educational intervention was conducted over a five-week period in a Digital Electronics course at the National University of Colombia. Through analyzing students' interactions with the tool and their perceptions of its usage, the study examined their learning experiences. Among the features highlighted by students as most beneficial for their HDL learning process were the online availability of the tool, the feedback system that helped them identify and correct errors in their code, the provision of immediate feedback, the online editor with syntax highlighting, and the graphical user interface. This work makes two significant contributions to the field of HDL teaching in engineering. Firstly, a publicly accessible HDL grading tool has been developed, offering students immediate formative and summative feedback through an automated grader. Secondly, empirical evidence has been provided regarding the benefits of using such a tool in enhancing students' learning process.

Keywords: Hardware Description Language (HDL), Automated grading, Feedback, Digital design learning, Learning of digital electronics

Introduction

In today's society, technology plays a fundamental role, and much of the technological advancement witnessed in recent decades can be attributed to the development and utilization of semiconductor materials in electronic components (Teepe, 2014). Such

components are present in modern electronic devices, including those used for telecommunications, transportation, medicine, entertainment, and numerous other applications (Teepe, 2014). Transistors serve as the fundamental device in electronic components based on semiconductor materials. These devices, combined in various configurations, have enabled the creation of digital electronic circuits (Corsini & Rizzo, 1991). These circuits are composed of logic gates and flip-flops, and their configuration can be defined by the graphic connection of components or through the use of a Hardware Description Language (HDL) (Corsini & Rizzo, 1991). Once the internal structure of the circuit is defined, it can be synthesized to obtain a map of logic elements that can be implemented in an integrated circuit or in a programmable logic device such as a Field Programmable Gate Array (FPGA) or a Complex Programmable Logic Device (CPLD).

HDLs are widely used in the hardware development industry due to their ability to simplify the design of highly complex circuits (Greenwood, 2009). Moreover, the graphical configuration of circuits can often be a cumbersome task, making the description of behavior through an HDL a preferred choice for many designers. Among the HDLs used in industry and education, Verilog and VHDL (Very High-Speed Integrated Circuit Hardware Description Language) stand out as the most widely utilized (LaMeres, 2019).

Therefore, it is essential for engineering students to acquire knowledge in creating electronic circuits using HDLs. Typically, these skills are acquired in universities that offer training programs in Electrical Engineering, Electronics, and Computer Science, and related fields (Corsini & Rizzo, 1991; Jutman et al., 2002; Madanayake et al., 2012; Pereira et al., 2012). Within these programs, it is common to find courses that focus on digital design, covering foundational concepts of logical circuits and their application in the creation and verification of digital electronic circuits using HDLs such as Verilog and VHDL. Educational resources such as the guide provided by the IEEE Computer Society and the ACM (Association for Computing Machinery) facilitate the design of curricula in the field of digital design and provide additional support (Impagliazzo et al., 2016).

In this context, it is worth noting that difficulties have been identified in learning the fundamentals of digital design, particularly concerning the use of software that is not tailored to educational processes for teaching HDLs, despite its prevalence in industry (Čirka & Kalúz, 2017). This mismatch between educational and industry software tools poses challenges for both teachers and students. Furthermore, limited access to and availability of laboratories often restrict students from utilizing necessary tools and receiving adequate support from instructors or teaching assistants for practical activities (Muchlas & Novianto, 2016). Additionally, students require continuous feedback to assess the correctness and improvement potential of their proposed solutions to digital design problems. However, due to time constraints, teachers often struggle to provide timely feedback on students' work within short periods (Baneres et al., 2014). Such circumstances

can hinder the learning process of students (Carroll, 2011). Therefore, to enhance the teaching of HDLs, educational institutions should offer students greater support through tools and resources designed to overcome these challenges.

In this sense, this work aims to address the identified needs and challenges associated with teaching and learning the fundamentals of digital electronics using HDLs. To achieve this, this paper presents an automated assessment software tool with feedback process to support the learning of HDL. In addition, an educational research study is conducted to analyze students' interactions with the tool and gain insights into their perceptions of the experience. The research question driving this study is: How do students interact with the automated grading software tool for their HDL designs, and what is their perception of the learning process? The proposed tool will serve as an alternative to proprietary software, enabling the automated grading of digital designs using HDL, providing immediate feedback on code errors, and allowing remote access, thus enhancing the learning experience for students.

This document is structured as follows: the second section provides a background and related works, the third section shows the development of the tool, the fourth section presents the design of the educational research, the fifth section shows the results of the study, the sixth section presents the discussion of the results, and the final section is about conclusions and future work.

Background and related works

Several initiatives have been developed to address the needs and challenges identified in teaching and learning the fundamentals of digital electronics and HDLs. Traditionally, the basics of digital electronics, including logic gates and their connections, have been taught using graphical representations. However, researchers have developed tools to teach circuit design using the same graphical representation. Mateev et al. (2004) developed a tool that compares small logic circuits with a Boolean expression to help students understand the transformation of a logical function into its graphical representation. Robal and Kalja (2007) created the e-EDU tool that teaches fundamental concepts of digital circuits and provides feedback to students by allowing them to manipulate component inputs and observe corresponding output behaviors. Furthermore, this tool is accessible online, eliminating limitations related to time and distance. However, these tools do not include automated grading functionality for circuits designed by students. This limitation is a significant drawback of widely-used teaching tools in digital electronics, including Hierarchical Computer Architecture design and Simulation Environment (HASE), Hardware Description Language implemented with Java (JHDL), Integrated Synthesis Environment WebPack (ISE WebPACK), Processor SIMulator (PSIM), Logisim, Virtual Vulcan, and e-learning environment e-EDU (e-EDU) (Stanisavljevic et al., 2013). The lack of

automated grading poses challenges, as the manual grading process can be time-consuming, and students often do not receive timely feedback on their work. Additionally, evaluating complex logic circuits is laborious, particularly when each student presents a different solution for each exercise.

To address these limitations, Stanisavljevic et al. (2013) developed a proprietary tool that compares the simulation output of the student's design with the expected output of the assigned exercise. However, this tool requires installation on each student's computer. Baneres et al. (2014) followed a similar approach by expanding the functionalities of the LogiSim tool (Burch, 2002). Their approach allows for the assignment of exercises to students and reports input values that caused problems in the designed circuit. The automated grading feature enables students to test their designs at any time and receive feedback on their errors without relying on laboratory availability or direct assistance from the teacher. Furthermore, Roy, Ghosh et al. (2015) introduced a web-based tool for creating, simulating, and analyzing logic circuits graphically. This tool incorporates automated grading of exercises through simulation or formal verification. Additionally, it offers the capability of dynamically generating additional activities to complement the exercises created by the teacher.

For HDLs, several tools have been developed specifically to support the learning of these languages. Nutter et al. (2014) created an automated grading process for circuits designed using graphical representations and HDLs. This process compares the student's proposed circuit with various circuits stored in a database. If the student's design exactly matches one of the registered circuits, satisfactory feedback is sent via email. In cases where the design does not match any circuit, a manual evaluation is performed to determine if the design meets the specified requirements. If it does, the design is added to the database as a new record. Other works include formal verification in addition to logical equivalence checking for grading designs created with HDLs. New Symbolic Model Verifier (NuSMV) (Cimatti et al., 2002) is a widely used tool for this purpose, using binary decision diagrams and a Boolean Satisfiability Problem (SAT) solver to perform symbolic verification of models. SAT solvers are used in several tools to compare the student's circuit with the circuit provided by the instructor. One tool that incorporates this functionality is presented by Petit et al. (2018) on the website jutge.org. This tool serves as a virtual judge to support the teaching of Verilog. The platform contains various courses with a set of exercises for learning digital design with HDLs. The feedback from this tool provides counterexamples to indicate cases where the student's circuit does not produce the same output as the instructor's circuit. However, this particular work does not explicitly address other types of errors, such as syntax errors. Based on an educational experience with this tool, the authors observed that students used it for additional hours and achieved greater productivity compared to scenarios with two or three instructors. Additionally, a survey

conducted at the end of the course showed a high level of student satisfaction. However, the authors point out that providing specific feedback on errors can lead to students losing the ability to simulate and find errors themselves.

Based on the literature review, it is evident that most existing tools for grading digital logic designs focus primarily on graphical applications rather than on the use of HDLs. Furthermore, these resources often provide binary verification of whether the student's solution is correct or incorrect, without detailed feedback to improve performance. In addition, the software used to design digital logic circuits with HDLs is not specifically designed for educational purposes, which poses a challenge to student learning. To address these limitations, this work proposes the development of an online tool that can be accessed at any time, offering a range of features to support skill development in digital logic design. These features include design simulation and testing, automated grading, immediate feedback without the need for teacher intervention, and progress tracking in the learning process. By incorporating such a tool, teachers can spend more time on topics that students find challenging and less time manually grading assignments. In addition, this type of tool would be particularly beneficial for self-study by students, providing them with a supportive learning environment (Stanisavljevic et al., 2013). By combining the advantages of online accessibility, automated grading, immediate feedback and progress tracking, the proposed tool aims to improve the teaching and learning experience of digital electronics and HDLs. It addresses the identified limitations of existing tools and seeks to provide a comprehensive resource that empowers students to develop their skills and supports teachers in effectively guiding their learning process.

UNCode-Digital Auto-Grader

The design and implementation of the UNCode-Digital Auto-Grader tool followed a prototype-based development strategy. This approach, as described by Pomberger et al. (1991), is an iterative and non-linear process aimed at achieving rapid results. The first step was to analyze and define the requirements of the tool based on an informal description of the users' needs. A prototype was then built and evaluated by users to identify any deficiencies and refine the requirements. The evaluation of the prototype included three types: exploratory, experimental, and evolutionary. The exploratory evaluation focused on defining the initial requirements, the experimental evaluation aimed to validate the specifications and architecture, and the evolutionary evaluation facilitated the incremental development of the system. The following sections provide a detailed description of the phases involved in the design of the tool in the context of this work.

Analysis and definition of requirements

To learn HDLs effectively, a solid understanding of the fundamentals of digital electronics is essential. While existing tools have focused primarily on graphical circuit design, they have provided valuable insight into learning these fundamentals. Based on the existing tools and the expertise of the authors, the requirements for developing a tool to support HDL learning were carefully analyzed and defined as follows:

- Continuous practice: Students need regular practice to reinforce their understanding (Mateev et al., 2004). Therefore, the tool should facilitate continuous exercises that allow students to constantly apply their knowledge.
- Timely feedback: Feedback is critical for effective learning (Baneres et al., 2014). The tool should provide automatic and timely feedback to students, allowing them to identify and correct errors in a timely manner. This feedback mechanism should reduce the burden on teachers by automating the assessment process (Nutter et al., 2014).
- Accessibility and convenience: The tool should be accessible online to eliminate time and distance constraints (Robal & Kalja, 2007). In addition, the user interface (UI) and user experience (UX) should be intuitive and user-friendly to enhance the learning process (Baneres et al., 2014).
- Web-based solution: The tool should be web-based so that students do not have to install additional applications on their computers. Assigned exercises should be solvable directly from a web browser using an online code editor that supports syntax highlighting.

In summary, the requirements for the HDL learning support tool include the ability to solve digital electronics problems, automated grading of student exercises, continuous feedback on both code and design, 24/7 online accessibility, and an online code editor with syntax highlighting capabilities.

Tool framework

The designed tool follows a web-based framework consisting of two main components: the front-end and the backend. Figure 1 illustrates the architecture of the HDL learning support tool. The frontend is the user-facing part of the system. It presents the digital electronics problem to be solved, provides an online editor with syntax highlighting for students to submit their proposed solutions, provides graphical feedback through timing diagrams, reports compilation errors and discrepancies with the reference model, and displays the quantitative grade for the proposed solution. The feedback is visualized in timing diagrams, which graphically represent the input and output signals of a circuit over time (LaMeres, 2019). If one of the outputs of the student's circuit does not match the expected behavior,

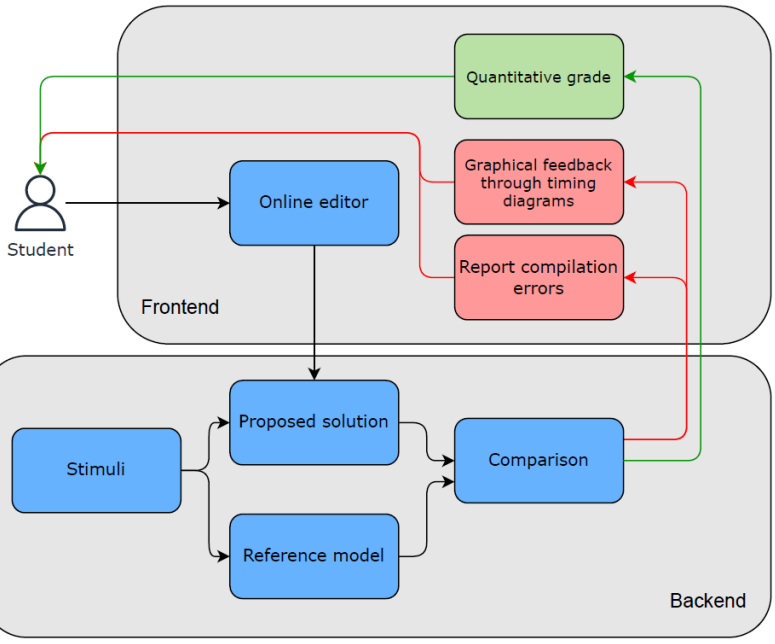

Fig. 1 Architecture of the HDL learning support tool

both the output of the student's circuit and the output of the reference model (a correct solution) are displayed.

The backend handles data manipulation and evaluation of student code and provides feedback accordingly. Three methods are typically used to evaluate HDL-based digital electronics solutions: formal verification, logical equivalence, and simulation-based verification. Formal verification involves transforming the design into mathematical models to test the correctness of the code through logical reasoning, often requiring significant computational resources. Logical equivalence is achieved by comparing designs using symbolic simulations with licensed tools. Simulation-based verification, which is used in this work, applies a set of stimuli to both the student's design (code) and a reference model to compare their results and identify differences (Fujita et al., 2008). Simulation-based verification requires two components: the design, which contains statements that define the behavior of the circuit, and a testbench file, which executes the simulation by invoking the design files. The testbench directs the application of stimuli to the inputs of the design at different times and records the output values (LaMeres, 2019). These stimuli are applied to both the reference model and the student's proposed solution, and their outputs are compared. If the simulation of the student's code encounters errors, they are reported. If the simulation finishes without errors, the outputs of the two designs are compared. If they match, the comparison block indicates that the student's code satisfactorily solves the problem. If there are discrepancies, the comparison block provides feedback on the input values where the student's design differs from the reference model.

Prototypes of the tool

Two prototypes were developed, one experimental and one evolutionary. The experimental prototype addressed three of the requirements: (1) it enabled the compilation and simulation of code written in Verilog and VHDL HDLs, (2) it compared the output results of the student's code with those of the reference model using predefined stimuli, and (3) it presented the comparison results in the form of a timing diagram. Moreover, the evolutionary prototype met the remaining requirements, including automatically grading and providing feedback on the student's design, and providing continuous web-based access without the need for additional software installation on the student's computer.

Experimental prototype

The first requirement of the experimental prototype was met by using open source software tools capable of compiling, synthesizing, and simulating HDL code. Instead of relying on tools associated with specific hardware vendors, the application aimed for independence. Specifically, Icarus Verilog (https://github.com/steveicarus/iverilog) was used for Verilog and GHDL (https://github.com/ghdl/ghdl) for VHDL. Compilation and simulation results were delivered in Value Change Dump files (VCD files).

To address the second requirement, which was to compare the results of the student code with those of the reference model using known stimuli, a testbench file was created. This file contained the necessary code to generate stimuli for both the student code and the reference model, along with code to observe the results. A web application was developed using PHP and JavaScript to facilitate this process. It had three text fields for entering the student's design, the testbench, and the reference model. The comparison between the output of the student's code and the reference model was done by analyzing the VCD files generated by the simulators. The Verilog_VCD package of the Python programming language (https://pypi.org/project/Verilog_VCD) was used to interpret these files. This package allowed the analysis of simulation output files and the transformation of the data into a Python dictionary, which facilitated the specification of timing diagrams.

To address the third requirement of displaying the comparison results using timing diagrams, several software options for visualization were evaluated. The Gigawave viewer (https://www.syncad.com/vcd_waveform_viewer.htm) and GTKWave (https://github.com/gtkwave/gtkwave) were considered but discarded due to the need to install additional software on the student's computer. Instead, WaveDrom (https://wavedrom.com) was selected (Chapyzhenka & Probell, 2016). WaveDrom, an open source tool with an MIT license, was written in JavaScript and allowed the generation of timing diagrams from a JSON format file called WaveJSON. WaveDrom is easily integrated into web applications. Comprehensive documentation for the project is available

at https://wavedrom.com, and an online editor for testing WaveJSON and viewing its graphical representation is available at https://wavedrom.com/editor.html.

The simulation data in Python was used to automatically generate the WaveJSON specification for the timing diagrams of both the reference model simulation and the student code being evaluated. By comparing these two diagrams, a new WaveJSON file was generated and its representation using WaveDrom was displayed in the web application. If there were no errors in the student's proposed solution, the behavior of all signals was displayed. However, if errors were detected, the obtained and expected signals were displayed, with moments of signal divergence highlighted in red. Figure 2 shows an example of a timing diagram obtained with the experimental prototype, with two output signals, F and G, along with their corresponding expected behavior, F* and G* (reference model). In the case of a single-bit output (e.g., F in the example), instances where the student's proposed design output signal (F) differed from the reference model output signal (F*) were highlighted in red. For a data bus output (e.g., G in the example), the entire value of the bus was highlighted in red to indicate a deviation from the expected value.

For better understanding, two Graphics Interchange Format (GIF) files have been created to show the Verilog and VHDL implementations of the experimental prototype. These files can be found at the following links:

- Verilog:
  https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/7d87d640d6bdb37d717e476ee43c4f7f4ee99bcd/gifs/verilog_v0.gif
- VHDL:
  https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/7d87d640d6bdb37d717e476ee43c4f7f4ee99bcd/gifs/vhdl_v0.gif.

The code developed for the implementation of the experimental prototype is available in the GitHub repository at: https://github.com/andrescorso/UNCode-Digital_V0.

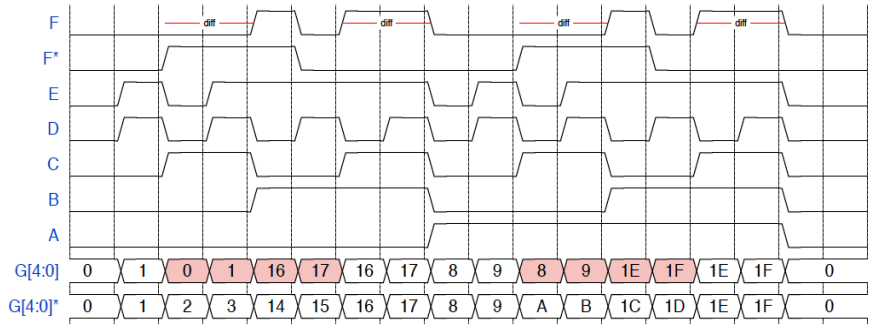

Fig. 2 Example of a timing diagram achieved with the experimental prototype

Evolutionary prototype

The evolutionary prototype was developed based on the experimental prototype, incorporating the requirements for automated grading and feedback of the design proposed by the student, as well as providing access to the web platform without the need for additional software installation on the student's computer. To achieve this, the UNCode tool was used. UNCode is an educational platform originally designed for programming courses at the National University of Colombia, specifically for the automated evaluation of programming exercises. The platform allows students to submit multiple attempts to solve programming problems in the form of source code or Jupyter notebooks. For each solution attempt, UNCode provides automated feedback through verdicts related to syntax, semantics, and program efficiency. It also assigns a numerical grade based on the test cases that the program successfully passes. UNCode provides several tools to support learning, including syntax highlighting, linter (programming best practice suggestions), code execution visualization, custom tests, and grading reports. Detailed information on the functionalities of UNCode can be found in the works of Restrepo-Calle et al. (2018, 2020). Due to the similarities between hardware description languages (HDLs) and programming languages and considering that UNCode already uses a virtual judge for the automatic evaluation of programming codes, it was considered appropriate to use this tool as the basis for the development of the evolutionary prototype.

To develop the evolutionary prototype, the UNCode tool was extended with a new functionality called UNCode-Digital, specifically designed to support HDLs. The architecture of UNCode-Digital is depicted in Figure 3. UNCode itself is based on INGInious (Roy, Derval et al., 2015). In the architecture diagram, the blue components represent the necessary additions and modifications made to create UNCode-Digital. The documentation for INGInious can be accessed at http://inginious.readthedocs.org. The developer documentation for INGInious served as the foundation for integrating the developments made in this research into UNCode, by creating additional plugins that can be added to the existing project without modifying the core code. The source code for UNCode is available in the GitHub organization https://github.com/JuezUN. UNCode is accessible to users through the portal https://uncode.unal.edu.co, where teachers can create exercises and students can solve them.

Figure 3 shows the incorporation of a reference model and the student's code, as well as the ability to provide feedback through a timing diagram. To implement this tool, a container was created that included the necessary compilers to simulate Verilog and VHDL code. In addition, an existing container was modified to include the necessary instructions for running simulations and comparing designs. In addition, a feedback plugin was integrated to display the timing diagram if there were any errors in the proposed solution. To meet the requirements of these new functionalities, adjustments were made to the task

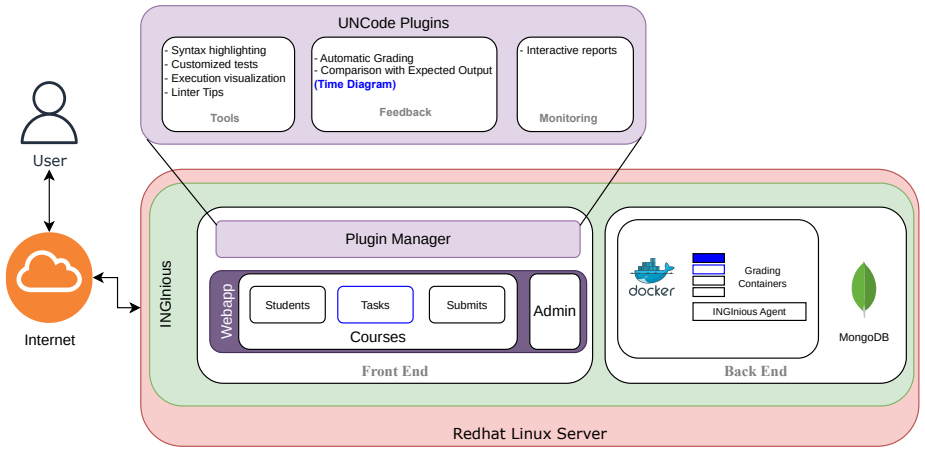

Fig. 3 Modified architecture of UNCode adding UNCode-Digital for automated evaluation of HDL codes

creation process within the tool. Finally, a modification was implemented to analyze the text and provide additional feedback to the student.

Figure 4 illustrates the feedback provided by UNCode-Digital as a student attempt to solve a problem using VHDL. To view the animated version of Figure 4, please visit the following link:

https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/master/gifs/UNCode_Digital/calificacion_tarea.gif.

The following links contain animated images with additional information about the functionalities achieved with the evolutionary prototype of this work:

- For the reference model and testbench creation interface:
  https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/master/gifs/UNCode_Digital/mod_creacion.gif

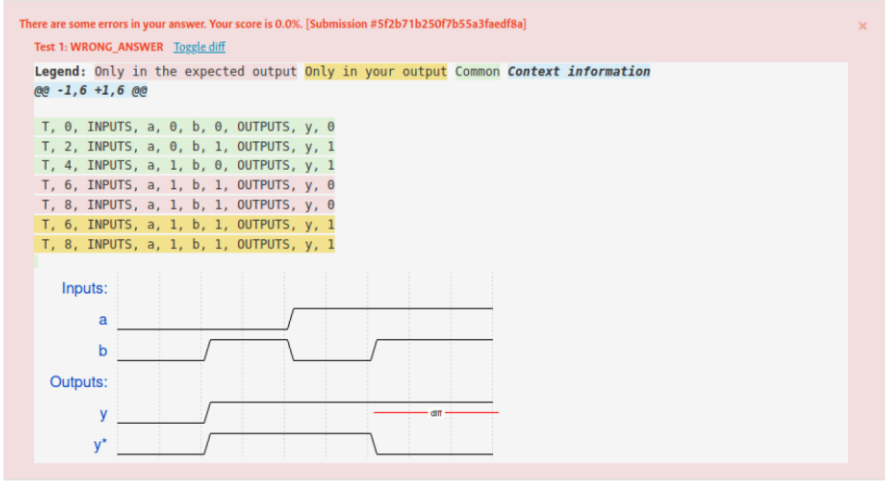

Fig. 4 Example of feedback provided by UNCode-Digital to the student

- On the interface for the creation of tasks by the teacher: https://github.com/andrescorso/UNCode-Digital_V0/blob/master/gifs/UNCode_Digital/creacion_tarea.gif
- On the feedback given by the tool to the student: https://github.com/andrescorso/UNCode-Digital_V0/blob/master/gifs/UNCode_Digital/uso_estudiante.gif

Design of the educational research

The educational research aimed to analyze students' interaction with UNCode-Digital and to explore their perception of the learning process in HDLs using this tool. The research followed a convergent design in which both quantitative and qualitative data were collected.

Course and setting

The Digital Electronics course, offered by several engineering programs at the National University of Colombia - Bogotá Campus, introduces students to the use of Hardware Description Languages (HDLs). The primary focus of this course is to provide students with a solid foundation in digital electronics, including topics such as numerical representations in different bases, combinational logic, sequential logic, and finite state machines. In the laboratory component of the course, students are engaged in the design and implementation of digital circuits. Laboratory sessions are held once a week for two (2) hours. While students primarily use the Verilog language to describe the hardware of their circuits, the UNCode-Digital platform also provides support for VHDL. The work described in this paper was conducted during the first academic semester of 2020. Due to the Covid-19 pandemic, all educational activities during this period were conducted remotely and synchronously over a period of six (6) weeks.

During this experience, practice exercises were designed to guide students from the most basic concepts to more advanced levels of difficulty. The professor started with an introduction to the topic through videos and then posed exercises for each student to solve using UNCode-Digital. Throughout the course, a series of exercises were designed to progressively guide students from basic concepts to more advanced levels of difficulty. The instructor initiated the learning process by providing introductory educational materials on the topics, followed by assigning exercises for each student to complete using the UNCode-Digital platform. In addition, detailed explanations were provided highlighting the essential language components required to develop each exercise. Students successfully completed exercises related to basic combinational logic circuits, with specific activities assigned on a weekly basis. The various topics and exercises covered during each week are summarized in Table 1.

Table 1 Topics in the Digital Electronics laboratory with descriptions of student exercises

Week	Theme	Exercises
0	Introduction to UNCode-Digital	Presentation of the tool with an exercise and the first submission of a circuit by the students
1	Introduction to Verilog	Design an XOR gate with two inputs and one output. Learn the different ways to describe a circuit in Verilog: structural, functional, and procedural
2	Logic circuit reduction	Three exercises on reducing logic circuits using Boolean algebra and coding the result using the descriptions learned in Week 1
3	Karnaugh maps	Four exercises using different numbers of variables to obtain the circuit by max-terms and min-terms. Check with UNCode-Digital to make sure that the minimum number of groupings was used in each case
4	Adder circuits	Designing a half adder, a full adder, and a four-bit adder using the modularity feature of HDLs
5	Conditional statements	Two exercises focused on using conditional statements in the HDL code to display a word on a 7-segment display using the if/else statement and to encode the alphabet using the case statement

Participants

The participants in this study were students enrolled in the Digital Electronics course at the National University of Colombia. Participants were selected using convenience sampling, and eligibility was based on their active engagement with the UNCode-Digital platform, specifically by making at least one submission. A total of 57 students interacted with the tool and were included as participants in the study. The participants represented a variety of university programs, with 23 students from electrical engineering (40.35%), 21 from electronic engineering (36.84%), 11 from mechatronic engineering (19.30%), and 2 from mechanical engineering (3.51%).

Measurement instruments

Quantitative data about students' interaction with the UNCode-Digital tool was automatically collected by the tool each time they submitted their code for evaluation. The data collected included information on the total number of participants in each exercise, the number of correct and incorrect code submissions, and the number of code submissions made by each student for each exercise. These data provided insight into student performance and engagement with the tool.

To gather qualitative data on student perceptions, an online survey was administered using the Google Forms platform. Prior to completing the survey, students were provided with an informed consent form that explained the purpose of the research, assured them of the confidentiality of their responses, and emphasized the voluntary nature of their

participation. The consent form also requested their explicit permission to use their responses in the study.

The survey included questions designed to capture students' experiences and perspectives regarding their learning process with UNCode-Digital. Each question was answered using a 6-point Likert scale: (1) Strongly disagree, (2) Disagree, (3) Slightly disagree, (4) Slightly agree, (5) Agree, and (6) Strongly agree. In addition, students were asked to provide a rationale for their chosen level on the scale. The questions were as follows:

The following questions refer to your experience using the UNCode-Digital tool for the development of Digital Electronics laboratories:

Statement 1: The automated grading provided by UNCode-Digital during the Digital Electronics lab was helpful in my learning of the Verilog hardware description language. *Why?*

Statement 2: The automated feedback using timing diagrams provided by UNCode-Digital during the Digital Electronics lab helped me identify errors in my designs and analyze how to correct them. *Why?*

Statement 3: The automated feedback using color-highlighted text provided by UNCode-Digital during the Digital Electronics lab helped me identify errors in my designs and analyze how to correct them. *Why?*

Statement 4: I find the online availability of UNCode-Digital to be a significant advantage. *Why?*

Statement 5: UNCode-Digital was useful in my learning process of the Verilog hardware description language. *Why?*

By using a Likert scale, the researchers aimed to capture the extent to which students agreed or disagreed with each statement, allowing for a quantitative representation of their perceptions. The inclusion of open-ended justifications for their responses allowed students to provide more detailed and qualitative insights into their experiences and the reasons for their chosen level on the scale. This combination of quantitative and qualitative data allowed for a comprehensive understanding of students' perspectives on the effectiveness and effect of UNCode-Digital in their learning process.

In addition to the Likert scale questions, the survey also included three open-ended questions to gather more detailed feedback from the students. These questions were:

1. What could be improved about the UNCode-Digital tool used during the labs?
2. What aspects of the UNCode-Digital tool used during the labs would you highlight?
3. Do you have any other comments?

The survey itself can be accessed at the following link: https://forms.gle/BKNoJDwhLwAyrdrE7

Data analysis

The quantitative data collected on students' interactions with UNCode-Digital, as well as the Likert-type survey responses, were analyzed using descriptive statistics. Additionally, for the qualitative analysis of the open-ended survey responses, the thematic analysis technique was employed (Bryman, 2016). The unit of analysis was the opinion provided by each student in response to the open-ended questions. The analysis process consisted of three stages: open coding, axial coding, and relativization of results.

In the open coding stage, codes were assigned to relevant words or phrases identified by students in relation to UNCode-Digital. In the axial coding stage, related codes were grouped into categories, and groupings of categories were identified to classify them into higher-level abstractions called themes. This iterative process was initially conducted by each researcher, and final analysis results were reached by consensus. In each iteration, the coding and categorization process was carried out to provide a more comprehensive interpretation of the students' responses. Finally, in the relativization phase of the results, the meaning of the themes and categories was interpreted to address the research question. The qualitative analysis was based on the responses of 48 participants.

Results

Students interaction with UNCode-Digital

The students in the course actively engaged with UNCode-Digital by participating in a total of 14 exercises, including an introductory task. On average, 52 of the 57 enrolled students (91.22% participation rate) participated in each exercise. Among the participants, 53 students attempted to solve more than 7 exercises. In 8 exercises all students who attempted were successful, while in 6 exercises some students were unable to reach a correct solution.

Figure 5 shows the number of submissions made by all students in each exercise using UNCode-Digital. The nomenclature "T" denotes an exercise, "T1" denotes the week in which the exercise was given (week 1), and "T1_1" denotes the exercise number (exercise 1). In total, students made 4488 submissions through the tool. On average, each exercise received 68 submissions with a correct answer (indicated by the blue color), with a standard deviation of four (4). Notably, there were also a considerable number of submissions with wrong answers (indicated by the orange color) or compilation errors (indicated by the green color). This suggests that students used the tool repetitively to refine their solutions, benefiting from the feedback provided by UNCode-Digital. Importantly, this indicates that students were able to receive feedback on their proposed solutions independently, without direct assistance from the course instructor or teaching assistant.

The exercises with the highest number of submissions were T2_1 and T4_1. In exercise T2_1, some students had difficulty simplifying the Boolean equation and assigning an

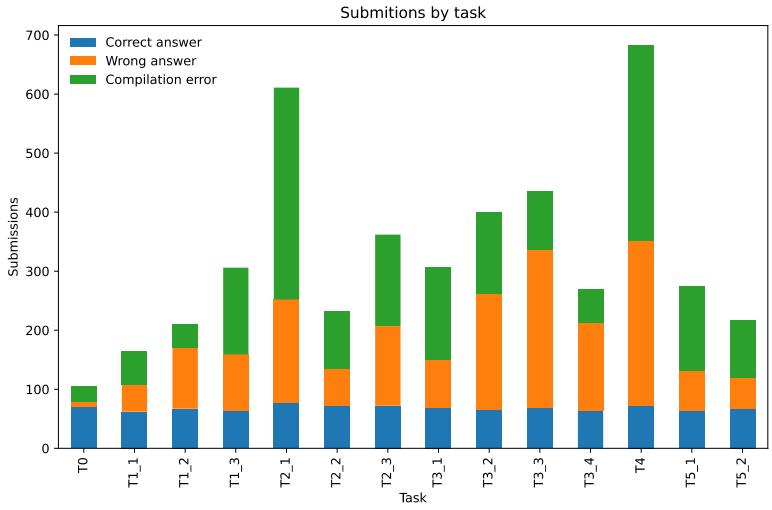

Fig. 5 Number of submissions made by all students using UNCode-Digital for each exercise proposed in the course

additional output to indicate the number of terms in the resulting equation (structural description). Exercise T4_1 was challenging for some students because it involved describing and interconnecting multiple Verilog modules, which was a new concept for them at the time.

Figure 6 shows the distribution of the total number of submissions made by each student during their attempts to solve each exercise. Results reveal that across all exercises, 50% of the students made fewer than 10 submissions (indicated by medians below 10). However, there are some outliers, particularly in exercise T2_1, where 6 students made more than 30 submissions. Remarkably, one student even made 80 submissions for this exercise.

Students' perceptions of their HDL learning processes using UNCode-Digital

Figure 7 shows the results of the students' Likert-scale responses regarding the level of agreement/disagreement for the first five questions of the perception survey.

Overall, the responses of the 48 students who participated in the survey indicate a high level of agreement regarding the usefulness of UNCode-Digital for learning HDLs. Specifically, 98.5% of the respondents agreed that the automated grading provided by the tool (Statement 1) and the ability to use the tool (Statement 5) were helpful in their HDL learning process. In addition, 91.7% of the students highlighted the support they received from the feedback provided through timing diagrams (Statement 2), while 89.6% agreed with the usefulness of the feedback provided through color-highlighted text (Statement 3). Furthermore, 100% of respondents emphasized the value of the tool being available online

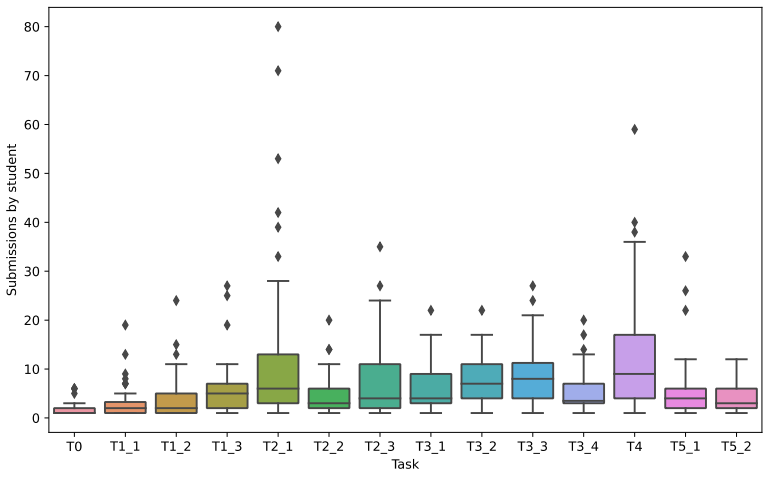

Fig. 6 Distribution of submissions made by students using UNCode-Digital for each exercise proposed in the course

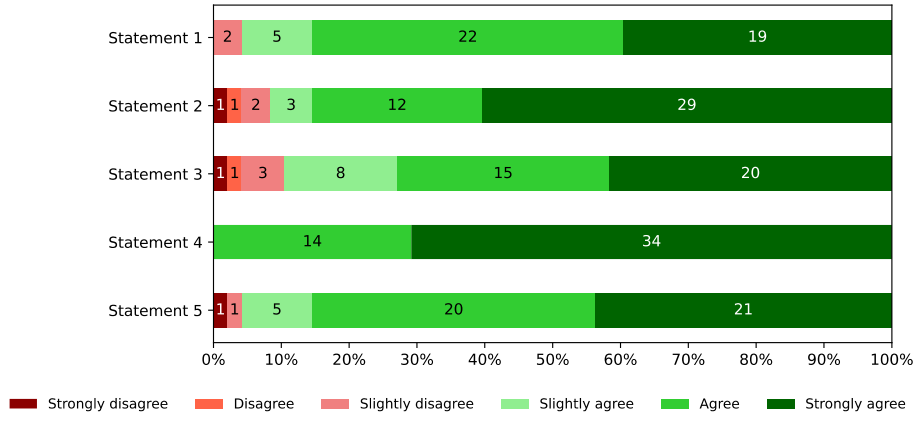

Fig. 7 Levels of agreement or disagreement of students with the statements made about their HDL learning experience using UNCode-Digital

(Statement 4). These results indicate that students have a positive perception of the various features and benefits of UNCode-Digital in supporting their learning experience of HDL.

Based on the qualitative analysis of the students' opinions from the open-ended questions, five themes emerged, listed in descending order of the number of comments made by the students:

1. *Perceptions of feedback:* Students found the feedback provided by UNCode-Digital to be valuable. They appreciated its usefulness in identifying and correcting errors in their code, its clarity, speed, and the ability to verify their designs through immediate

feedback. However, some students mentioned challenges in interpreting the feedback provided by the tool.

2. *Uses of UNCode-Digital for learning:* Students highlighted the various ways in which UNCode-Digital was beneficial to their learning process. They mentioned its role in facilitating a gradual introduction to HDLs, improving their skills by identifying and correcting errors, enhancing the didactics of the course, allowing for continuous practice, supporting autonomous learning during the Covid-19 pandemic, and offering advantages over other programming tools for HDLs.
3. *Perceptions of tool features:* This theme encompassed the positive user experience of using UNCode-Digital. Students appreciated the ease and speed of compiling and verifying solutions, the visual appeal of the tool, and the usefulness of the feedback provided by timing diagrams and color-highlighted text. They also noted the availability of features such as viewing history and multiple attempts. In addition, they recognized the benefits of an online tool, which eliminates the need for additional software downloads and provides accessibility at any time.
4. *Perceptions of automated grading:* Students recognized the immediate grading feature of UNCode-Digital and its ability to accurately grade multiple designs that met the assignment requirements. However, some students noted limitations of the automated grading, such as the tool's lack of consideration of intermediate grades for partially correct solutions.
5. *Perceptions of improvements to UNCode-Digital:* Students suggested several areas for improvement in UNCode-Digital. These included improving the presentation of errors, integrating documentation into the tool, indicating the exact location of errors in the code, incorporating file handling capabilities instead of a text editor, integrating with physical hardware, providing more examples, promoting its use in other courses, and adding more exercises within the tool.

Analysis of these issues provides valuable insight into the students' perspectives on UNCode-Digital, highlights its strengths and areas for improvement, and demonstrates the overall positive reception of the tool for learning HDLs.

Table 2 provides examples of student opinions categorized into the previously discussed themes, along with the corresponding number of opinions classified in each one. It should be noted that the number of opinions exceeds the number of participants in the study, as each response may encompass multiple situations related to the tool and thus be classified in multiple categories simultaneously.

Table 2 Examples of students' perceptions of the UNCode-Digital tool and the number of opinions classified in each theme

Themes	Examples of opinions	Number of opinions
Perceptions of feedback	"Obtaining a didactic and visual representation of the results allows us as students to relate the code we are executing with the error committed and it can be corrected and identified easily."	204
	"It is useful to contrast with the expected result and quickly see where in the code the error is."	
	"Although it helps in identifying errors, I think it is not clear enough."	
Uses of UNCode-Digital for learning	"UNCode-Digital is a very useful tool because it helps us practice and understand Verilog coding, as well as find errors and improve autonomously."	92
	"Because the self-grading feature allowed me to review my program and find my mistakes, so I could correct them and learn from them."	
Perceptions of tool features	"I think having a graphic guide of what it should give versus what it's actually giving makes it easier to correct errors."	73
	"I think it's a good tool because it's online and there's no need to download any files, so you can work from any computer."	
Perceptions of automated grading	"I was surprised that the platform could interpret multiple solutions that can be given to the same exercise."	37
	"It only graded 0% and 100% but I never saw it marking 25%, 50%, or 80%."	
Perceptions of improvements to UNCode-Digital	"Add information about basic concepts to start programming."	37

Discussion

An educational research study was conducted to evaluate the effectiveness of UNCode-Digital, an automated grading and feedback tool for learning HDLs. The study was conducted during the first academic period of 2020 in the Digital Electronics course at the National University of Colombia. The research aimed to investigate how students interacted with the automated grading software tool for their HDL designs and to examine their perceptions of the learning process.

Quantitative results showed that students consistently engaged with the tool to complete the assigned exercises. Analysis showed that the majority of students submitted a considerable number of submissions, with some students exceeding that number. This allowed the instructor to identify areas where students needed additional support or

reinforcement and provide tailored feedback accordingly. The comprehensive submission history also allowed for detailed analysis, as shown in Figure 5, where two students made a higher number of submissions for Task 4 than their peers.

Qualitative data collected through open-ended survey questions revealed five key themes that provide insight into the effect of the tool on student learning. A remarkable 95% of students found UNCode-Digital to be beneficial in their learning of the Verilog language. Students expressed that the learning environment created with the tool in their lab classes was highly beneficial, especially given the remote and synchronous nature of the experience during the Covid-19 pandemic. According to the students' feedback, the tool facilitated a gradual introduction to HDLs, enabled them to identify and correct errors in the exercises, promoted consistent practice, and supported their independent work.

According to the students' feedback, the UNCode-Digital tool was highly valued for its ability to facilitate the process of finding and correcting code errors in a convenient, fast, and clear manner. This positive trend contrasts with the partially positive results reported in the study conducted at the University of Manchester by Nutter et al. (2014). It is important to note that the tool used at the University of Manchester provided feedback only once per lab, the exercises were more complex, and it also assessed students' testbenches. In contrast, the students in this study appreciated the benefit of receiving feedback multiple times during each exercise, which they perceived as beneficial to their learning process. In addition, over 90% of the students found one of the two types of feedback (timing diagram or text highlighting) useful for error identification and analysis. However, some students suggested that the feedback could be improved by making it clearer and more explicit, and by including intermediate grades.

In addition, students expressed satisfaction with the usability and speed of the tool, highlighting its visual design, the availability of two types of feedback, the ability to make multiple attempts, and the ability to review their progress through the course. Notably, all students agreed that the online availability of the tool was a substantial advantage, as it allowed them to access the tool from anywhere at any time without the need for additional software. These comments are consistent with the findings of Baneres and Saíz (2016), who reported difficulties with the installation of similar tools, which limited their use by some students.

Students also emphasized the value of the automated grading feature, as it allowed them to review multiple possible solutions to exercises and provided immediate feedback, eliminating the need to wait for a teacher's review. The majority (95%) of the students surveyed considered the tool's automated grading to be beneficial for their learning of HDLs, which is consistent with the results of the study conducted by Baneres and Saíz (2016).

In addition, students provided suggestions for improving the tool, including both modifications to existing elements and the addition of new features. One common suggestion was to improve the clarity of the feedback, which addressed a concern raised by students who had previously used similar tools in other studies (Baneres & Saíz, 2016; Nutter et al., 2014). While students appreciated the feedback provided during lab exercises, they expressed a need for more information, clarity, and guidance to help them identify and correct errors in their HDL code. Moreover, students requested an increased number of examples and exercises, a recommendation that was also highlighted in the study conducted by Baneres and Saíz (2016). Finally, as suggested in Nutter et al. (2014), it is important to continue collecting data and refining the tool to increase student satisfaction.

Conclusions

This article presented UNCode-Digital, a software tool designed to help students learn Hardware Description Languages. The tool provides automated grading of HDL code and provides feedback features to enhance the students' learning experience. Two prototypes of the tool were developed, and a convergent educational study was conducted to evaluate its effectiveness and students' perceptions.

The study results showed that students had positive perceptions of UNCode-Digital and found it useful for their Verilog HDL learning. The automatic grading, feedback, online availability, syntax highlighting, and graphical interface were highlighted by the students as particularly useful features. The educational intervention using the tool provided an opportunity for the students to overcome the challenges posed by the Covid-19 pandemic and engage in productive learning activities.

This work contributes to the field of HDL education in engineering in two significant ways. First, it provides a publicly available HDL grading tool, UNCode-Digital, that provides immediate summative and formative feedback and assessment to students through automated grading. Second, it presents empirical evidence supporting the benefits of using such a tool in the learning process. Overall, the development of UNCode-Digital and the results of this study contribute to the advancement of HDL education and offer potential avenues for further exploration and improvement in the field.

Future work can focus on improving the tool based on the limitations identified by the participants. This includes providing more detailed feedback to facilitate error identification, incorporating hardware synthesis to enable testing on physical devices, and exploring alternative methods for comparing input and output signals, especially for larger projects. It is important to recognize that the results of this study are specific to its context, and further experimental research is needed to gain a deeper understanding of the effects of tools such as UNCode-Digital on various aspects such as academic performance, student engagement, motivation, and meaningful learning. For example, in order to assess the

effect of the automated grading software tool on student learning outcomes in HDL, one possible approach involves randomly assigning participants to either an experimental group using the tool or a control group without the tool. This approach also involves the administration of pre-tests and post-tests followed by statistical analysis. In this approach, participants are randomly assigned to either the experimental group or the control group. The experimental group uses the automated grading software tool as part of their learning experience, while the control group does not have access to the tool. Both groups take pre-tests to establish a baseline of knowledge and understanding prior to the intervention. Post-tests are then administered to measure any changes in learning outcomes. Following the collection of data from the pre-tests and post-tests, statistical analysis techniques will be used to evaluate the effect of the automated grading software tool. These analyses aim to identify any significant differences between the experimental and control groups, indicating whether the tool has affected student learning outcomes in HDL. Moreover, in future work, we recommend using either the Technology Acceptance Model (TAM) or the Unified Theory of Acceptance and Use of Technology (UTAUT) model to evaluate the automated grading software tool, allowing for a comprehensive assessment of user acceptance and adoption. Gathering student perceptions through questionnaires, interviews, or user feedback sessions will provide valuable insights to guide the further development of the tool and improve the learning experience in the field of HDL.

Abbreviations

ACM: Association for Computing Machinery; CPLD: Complex Programmable Logic Device; e-EDU: e-learning environment e-EDU; FPGA: Field Programmable Gate Array; GIF: Graphics Interchange Format; HASE: Hierarchical Computer Architecture design and Simulation Environment; HDL: Hardware Description Language; IEEE: Institute of Electrical and Electronics Engineers; ISE WebPack: Integrated Synthesis Environment WebPack; JHDL: Hardware Description Language implemented with Java; NuSMV: New Symbolic Model Verifier; PSIM: Processor SIMulator; SAT: Boolean Satisfiability Problem; TAM: Technology Acceptance Model; UI: User Interface; UTAUT: Unified Theory of Acceptance and Use of Technology; UX: User Experience; VCD: Value Change Dump files; VHDL: Very High-Speed Integrated Circuit Hardware Description Language.

Authors' contributions

All authors: Conceptualization, Methodology, Validation and Formal Analysis, Investigation, Writing Original and Draft Preparation, Reviewing and Editing, Data Curation, Supervision and Project Administration.

Authors' information

Andrés Francisco Corso Pinzón received the master science degree in Computation and Systems Engineering from Universidad Nacional de Colombia, Bogotá, Colombia, in 2023. Currently, he is a development engineer in a software development company in Colombia.

Jhon Jairo Ramírez-Echeverry received the Ph.D. degree (cum laude) in Engineering of Projects and Systems from the Polytechnic University of Catalonia, (Universitat Politècnica de Catalunya) - BarcelonaTech, Spain, in 2017. Currently, he is an Associate Professor in the Department of Electrical and Electronic Engineering at the Universidad Nacional de Colombia, Bogotá, Colombia. His research interests focus primarily on engineering education, with a special emphasis on self-regulated learning.

Felipe Restrepo-Calle received the Ph.D. degree (cum laude) from the University of Alicante, Spain, in 2011. He worked as a postdoctoral research fellow at the University of Seville, Spain, in 2012 and 2013. Since 2014, he has been working in the Department of Systems and Industrial Engineering at the National University of Colombia (Universidad Nacional de Colombia), Bogotá, Colombia, where he is an associate professor and leads the Programming Languages and Systems (PLaS) research group. His fields of research interest include programming languages, dependable design in embedded systems and engineering education.

Funding
Not applicable.

Availability of data and materials

1. GIF files have been created to show the Verilog and VHDL implementations of the experimental prototype. These files can be found at the following links:
   - Verilog: https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/7d87d640d6bdb37d717e476ee43c4f7f4ee99bcd/gifs/verilog_v0.gif
   - VHDL: https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/7d87d640d6bdb37d717e476ee43c4f7f4ee99bcd/gifs/vhdl_v0.gif.
2. The code developed for the implementation of the experimental prototype is available in the GitHub repository at: https://github.com/andrescorso/UNCode-Digital_V0.
3. Animated version of Figure 4 is available at the following link: https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/master/gifs/UNCode_Digital/calificacion_tarea.gif.
4. Animated images with additional information about the functionalities achieved with the evolutionary prototype of this work:
   - For the reference model and testbench creation interface: https://raw.githubusercontent.com/andrescorso/UNCode-Digital_V0/master/gifs/UNCode_Digital/mod_creacion.gif
   - On the interface for the creation of tasks by the teacher: https://github.com/andrescorso/UNCode-Digital_V0/blob/master/gifs/UNCode_Digital/creacion_tarea.gif
   - On the feedback given by the tool to the student: https://github.com/andrescorso/UNCode-Digital_V0/blob/master/gifs/UNCode_Digital/uso_estudiante.gif

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 19 May 2023 Accepted: 21 July 2023
Published online: 1 January 2024 (Online First: 4 August 2023)

References

Baneres, D., & Saíz, J. (2016). Intelligent tutoring system for learning digital systems on MOOC environments. In *Proceedings of 2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems* (pp. 95–102). IEEE. https://doi.org/10.1109/CISIS.2016.47

Baneres, D., Clariso, R., Jorba, J., & Serra, M. (2014). Experiences in digital circuit design courses: A self-study platform for learning support. *IEEE Transactions on Learning Technologies*, *7*(4), 360–374. https://doi.org/10.1109/TLT.2014.2320919

Bryman, A. (2016). *Social research methods (4th edition)*. Oxford University Press.

Burch, C. (2002). Logisim: A graphical system for logic circuit design and simulation. *Journal on Educational Resources in Computing*, *2*(1), 5–16. https://doi.org/10.1145/545197.545199

Carroll, C. R. (2011). Teaching ground-floor digital circuits to pre-engineering students. In *Proceedings of 2011 ASEE Annual Conference & Exposition* (pp. 22.1394.1–22.1394.8). American Society for Engineering Education. https://doi.org/10.18260/1-2--18411

Chapyzhenka, A., & Probell, J. (2016). *WaveDrom rendering beautiful waveforms from plain text*. Synopsys User Group (SNUG). https://wavedrom.com/images/SNUG2016_WaveDrom.pdf

Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., & Tacchella, A. (2002). NuSMV 2: An opensource tool for symbolic model checking. In E. Brinksma & K. G. Larsen (Eds.), *Computer Aided Verification. CAV 2002. Lecture Notes in Computer Science, vol 2404* (pp. 359–364). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45657-0_29

Čirka, L., & Kalúz, M. (2017). A web-based tool for design of Simulink models. In *Proceedings of the 2017 21st International Conference on Process Control* (pp. 92–97). IEEE. https://doi.org/10.1109/PC.2017.7976195

Corsini, P., & Rizzo, L. (1991). SSCSSC: A tool for the teaching of digital circuits. *IEEE Transactions on Education*, *34*(1), 70–75. https://doi.org/10.1109/13.79884

Fujita, M., Ghosh, I., & Prasad, M. (2008). *Verification techniques for system-level design*. Elsevier. https://doi.org/10.1016/B978-0-12-370616-4.X5001-0

Greenwood, G. W. (2009). Teaching hardware description languages to satisfy industry expectations. *International Journal of Electrical Engineering & Education*, *46*(3), 239–247. https://doi.org/10.7227/IJEEE.46.3.3

Impagliazzo, J., Conry, S., Durant, E., Hughes, J. L. A., Herger, L., Junlin, L., Lam, H., McGettrick, A., Reese, R., & Weidong, L. (2016). *Computer Engineering Curricula 2016.* Technical Report. ACM Press and IEEE Computer Society Press. https://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf

Jutman, A., Ubar, R., Hahanov, V., & Skvortsova, O. (2002). Practical works for on-line teaching design and test of digital circuits. In *9th IEEE International Conference on Electronics, Circuits and Systems* (Vol.3, pp. 1223–1226). IEEE. https://doi.org/10.1109/ICECS.2002.1046474

LaMeres, B. J. (2019). *Introduction to logic circuits & logic design with VHDL (2nd edition)*. Springer Nature Switzerland.

Madanayake, A., Wijenayake, C., Joshi, R. M., Grover, J., Carletta, J., Adams, J., Hartley, T., & Ogunfunmi, T. (2012). Teaching freshmen VHDL-based digital design. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 2701–2704). IEEE. https://doi.org/10.1109/ISCAS.2012.6271865

Mateev, V., Manoilov, P., & Iliev, M. (2004). Tutoring tool for logical schemes design. In *Proceedings of International Conference on Computer Systems and Technologies* (pp. I.9-1–I.9-5). Association for Computing Machinery. https://doi.org/10.1145/1050330.1050343

Muchlas, M., & Novianto, M. A. (2016). An online lab for digital electronics course using information technology supports. In *Proceedings of 2015 International Conference on Science in Information Technology: Big Data Spectrum for Future Information Economy* (pp. 299–302). IEEE. https://doi.org/10.1109/ICSITech.2015.7407821

Nutter, P. W., Pavlidis, V. F., & Pepper, J. (2014). Efficient teaching of digital design with automated assessment and feedback. In *Proceedings of 10th European Workshop on Microelectronics Education* (pp. 203–207). IEEE. http://doi.org/10.1109/EWME.2014.6877426

Pereira, M. C., Viera, P. V., Raabe, A. L. A., & Zeferino, C. A. (2012). A basic processor for teaching digital circuits and systems design with FPGA. In *Proceedings of 2012 VIII Southern Conference on Programmable Logic* (pp. 1–6). IEEE. https://doi.org/10.1109/SPL.2012.6211804

Petit, J., Roura, S., Carmona, J., Cortadella, J., Duch, J., Gimnez, O., Mani, A., Mas, J., Rodriguez-Carbonell, E., & Rubio, E. (2018). Jutge.org: Characteristics and experiences. *IEEE Transactions on Learning Technologies*, *11*(3), 321–333. https://doi.org/10.1109/TLT.2017.2723389

Pomberger, G., Bischofberger, W. R., Kolb, D., Pree, W., & Schlemm, H. (1991). Prototyping-oriented software development - Concepts and tools. *Structured Programming*, *12*(1), 43–60.

Restrepo-Calle, F., Ramírez-Echeverry, J. J., & González, F. A. (2018). UNCode: Interactive system for learning and automatic evaluation of computer programming skills. In *Proceedings of 10th International Conference on Education and New Learning Technologies* (pp. 6888–6898). International Academy of Technology, Education and Development. https://doi.org/10.21125/edulearn.2018.1632

Restrepo-Calle, F., Ramírez-Echeverry, J. J., & González, F. A. (2020). Using an interactive software tool for the formative and summative evaluation in a computer programming course: An experience report. *Global Journal of Engineering Education*, *22*(3), 174–185. http://www.wiete.com.au/journals/GJEE/Publish/vol22no3/06-Echeverry-J.pdf

Robal, T., & Kalja, A. (2007). Applying e-environments in teaching the basics of digital logic. In *Proceedings of 2007 IEEE International Conference on Microelectronic Systems Education* (pp. 41–42). IEEE. https://doi.org/10.1109/MSE.2007.24

Roy, G., Ghosh, D., Mandal, C., & Mitra, I. (2015). Aiding teaching of logic design and computer organization through dynamic problem generation and automatic checker using COLDVL tool. In *Proceedings of 2015 IEEE Seventh International Conference on Technology for Education* (pp. 15–22). IEEE. https://doi.org/10.1109/T4E.2015.4

Roy, P. V., Derval, G., Frantzen, B., Gego, A., & Reinbold, P. (2015). Automatic grading of programming exercises in a MOOC using the INGInious platform. In *Proceedings of the European MOOC Stakeholder Summit 2015* (pp. 86–91). Université catholique de Louvain and P.A.U. Education.

Stanisavljevic, Z., Pavlovic, V., Nikolic, B., & Djordjevic, J. (2013). SDLDS—System for Digital Logic Design and Simulation. *IEEE Transactions on Education*, *56*(2), 235–245. https://doi.org/10.1109/TE.2012.2211598

Teepe, G. (2014). The growing importance of microelectronics from a foundry perspective. *2014 Design, Automation & Test in Europe Conference & Exhibition* (pp. 1-1). IEEE. https://doi.org/10.7873/DATE.2014.015

Publisher's Note