AI-Assisted Learning of VHDL

Prof. Yumin Zhang, Southeast Missouri State University

Yumin Zhang is a professor in the Department of Engineering and Technology, Southeast Missouri State University. His research interests include semiconductor devices, electronic circuits, neural networks, and engineering education.

Dr. Bradley Deken, Southeast Missouri State University

Bradley J. Deken is a chairperson and professor in the Department of Engineering and Technology at Southeast Missouri State University. His research interests include programmable logic controllers, industrial automation, and engineering education.

AI-Assisted Learning of VHDL

Yumin Zhang, Brad Deken

Department of Engineering and Technology
Southeast Missouri State University
Cape Girardeau, MO 63701

Abstract

VHDL is widely used in digital systems design courses in electrical engineering programs, yet many students struggle with its steep learning curve. Integrating AI tools like ChatGPT presents an innovative and effective solution to these challenges. By leveraging AI in the learning process, students can benefit from real-time, interactive feedback on coding errors, logic flaws, and conceptual misunderstandings.

Student feedback suggests that AI tools are generally helpful, particularly for debugging syntax errors in VHDL code since EDA software often fails to provide precise error information. However, misuse of AI can have drawbacks. Some students have become overly reliant on AI by using generated code without fully understanding it. Consequently, when deep-rooted errors occurred, they struggled to identify and resolve the issues. The distribution of final exam scores highlights AI tools as a double-edged sword. While some students benefited from its use, others were negatively affected.

Introduction

AI tools are transforming the learning experience for engineering students. By acting as intelligent tutors, these tools provide personalized feedback, adaptive learning paths, and real-time support. For example, AI platforms can analyze a student's performance, identify weaknesses in their work, and recommend specific resources to help them improve. In hands-on courses, AI simplifies tasks like coding, simulation, and debugging to allow students to focus on developing crucial problem-solving skills. Additionally, AI fosters collaboration and inspires creativity by offering insights into innovative design and optimization methods. As a result, AI is making engineering education more accessible, efficient, and relevant to the skills students need for today's industry [1-3].

In modern digital systems design courses, Field Programmable Gate Arrays (FPGAs) have superseded discrete logic gates. In our department, most students take the digital system design course in their junior year and will already have completed a course electric circuit analysis. However, FPGA development can still be challenging for students. The proficiency required in hardware description languages (HDLs) like VHDL or Verilog and the complex synthesis and debugging procedures can be difficult for students to attain. Large Language Models (LLMs), such as OpenAI's ChatGPT, offer a promising solution by assisting with these complex tasks and streamlining the FPGA design process [4-6].

AI tools can help students learn digital system design in several ways. It can assist with conceptual understanding, code assistance and debugging, documentation, and optimization of

the design. For example, the tools can break down complex HDL concepts into digestible explanations by explaining concepts such as the difference between combinational and sequential logic circuits. In addition, they can generate example code snippets, explain the syntax, and provide templates for many common design patterns [7]. Furthermore, students often struggle with cryptic error messages from electronic design automation (EDA) tools. However, AI tools can often translate these messages into clearer explanations and suggest specific fixes for the errors [8]. For advanced applications, AI tools can assist in generating HDL testbenches by creating comprehensive test cases that cover various input scenarios [9]. This ensures that the hardware behaves as expected under various test conditions.

VHDL Syntax with AI Tools

VHDL is considered a strongly typed language because it enforces strict rules regarding data types and their usage. In VHDL, every signal, variable, and constant must have a defined type. Furthermore, operations between differing types often require explicit type conversion. This strong typing ensures that errors related to incompatible data types are caught during the compilation phase. This is important for improving code reliability and maintainability. By mandating clear and unambiguous definitions, VHDL reduces the likelihood of subtle runtime errors. This is particularly critical in hardware design where incorrect behavior can have significant consequences. This characteristic makes VHDL well-suited for designing complex and reliable digital systems. However, these characteristics also make VHDL difficult to learn for students.

Beginners often struggle with VHDL syntax due to its strict and formal structure that demands precise adherence to rules. Common mistakes include incorrect use of semicolons, misplacing keywords, and omitting required statements like "end process". Additionally, the requirement to explicitly define data types and the need for proper declarations for signals, variables, and constants can be confusing for new learners. Errors in process sensitivity lists and improper usage of concurrent versus sequential statements are also frequent pitfalls. These challenges can lead to compilation errors that might be intimidating for novices. However, with practice and a thorough understanding of VHDL's syntax and semantics, beginners can overcome these initial hurdles and develop robust hardware descriptions.

In addition, electronic design automation (EDA) software packages, such as Quartus from Altera and Intel, often provide vague or cryptic error messages. The lack of specificity can be particularly frustrating for beginners. These messages often point to a symptom of the problem rather than its root cause leading inexperienced designers to decipher the issue through trial and error. For example, an error might simply state "syntax error near line X" without indicating the specific issue even when the issue is as simple as a missing semicolon or an undefined signal. This lack of clarity can slow down the debugging process and increase development time. Additionally, error messages often fail to contextualize the problem within the broader design. This makes it difficult for users to understand how a local error affects the overall system. To mitigate this, users often rely on community forums, documentation, and experience to interpret and resolve these errors effectively.

AI tools, such as ChatGPT, can be a valuable tool for students debugging their VHDL code by providing quick and accurate insights into potential errors and their solutions. By analyzing the

code and identifying issues like syntax errors, incorrect data type usage, or misplaced constructs, ChatGPT can offer detailed explanations and suggest precise corrections. Students can also use ChatGPT to clarify VHDL concepts, learn best practices, or generate examples to understand complex topics better. Furthermore, ChatGPT's ability to assist with interpreting cryptic error messages from EDA tools can significantly reduce the time spent troubleshooting. By serving as a virtual mentor, ChatGPT helps students build confidence in their VHDL skills while reinforcing their understanding of digital design principles.

Integration of ChatGPT into the Digital Design Course

Within this work, ChatGPT was integrated into a digital systems design course that is taken by students in engineering and engineer technology programs. This course introduced students to many digital systems topics including logic circuit elements, logic functions, Boolean algebra, combinational logic circuits, finite state machines, and memory devices. The course includes an extensive lab component and programmable logic devices (PLD) and their programming with VHDL feature prominently.

Within the course students were introduced to the use of ChatGPT to assist with VHDL lab tasks. They were shown examples of how to use it as a tool to assist with debugging of VHDL code and correction of errors. They were also shown how to ask ChatGPT to clarify concepts related to the course content. This content was provided at the beginning of the semester and then it was left up to the student to decide the extent to which they used ChatGPT.

Towards the end of the semester, students were assigned a survey as a homework task. Of the fourteen students enrolled in the course, twelve responded. Although the sample size is small, the results still provide insight into student perceptions. The first question asked how AI tools were used in the course. Every student reported using ChatGPT, while only a few mentioned Google Gemini. Then, the survey asked questions about the student's perceptions of the use of AI tools. Table 1 presents the results of the following three questions:

Q2: How would you rate the clarity of explanations provided by ChatGPT for VHDL concepts?
Q3: To what extent did ChatGPT's responses help you in understanding and applying VHDL design principles?
Q4: Did ChatGPT provide accurate and relevant assistance in debugging VHDL code and identifying errors?

Table 1 Survey Result 1

Q2	**Very Clear**	**Clear**	**Neutral**	**Unclear**	**Very Unclear**
		7	**3**	**1**	**1**
Q3	**Very Helpful**	**Helpful**	**Neutral**	**Slightly Helpful**	**Not Helpful**
	1	**6**	**2**	**1**	**2**
Q4	**Always**	**Most Time**	**Occasionally**	**Rarely**	**Never**
		9	**2**		**1**

The survey results indicates that most students considered ChatGPT to be very helpful. A clear exception to this was a single student who did not use it much and provided the most negative

reviews. During lab sessions in the first half of the semester, students used ChatGPT to resolve most syntax issues. For comparison, in previous years students struggled a lot more with these issues early in the semester. Additionally, a few students developed a strong interest in the topic. One student stated that this was the most interesting course in the entire curriculum due in large part to this use of AI. The instructor also noted much less frustration with syntax issues within the classroom.

Misuse of AI Tools

While ChatGPT can be an excellent resource for learning and debugging VHDL code, it is sometimes misused by students who rely on it excessively to generate complete code without fully understanding the underlying concepts. This over-reliance can lead to superficial learning where students bypass the critical process of problem-solving and design thinking. Without these essentials for mastering hardware description languages, students are only learning how to use ChatGPT and not learning how to design digital systems. Moreover, if students fail to verify or critically evaluate the code generated by ChatGPT, they risk introducing errors or inefficiencies into their designs. Misusing ChatGPT as a shortcut rather than as a learning aid can undermine their ability to develop the independent skills needed to tackle real-world hardware challenges. To maximize its benefits, students should use ChatGPT as a supplementary tool while actively engaging with the principles and practices of VHDL coding.

For example, combinational logic is covered in the first half of the semester, and students are supposed to write VHDL code in this style. However, the lab reports from some students indicated that sequential style VHDL codes were used that were generated from ChatGPT. In the course, the seven-segment display was used extensively. The lab kit we used (DE0-CV) has active-low inputs for its LED segments. When the students asked ChatGPT to generate the VHDL codes, active-high inputs was assumed by the AI. Therefore, their VHDL codes were not able to display the correct numbers.

In the survey mentioned above, the following three questions were also asked on the downsides of using ChatGPT. The results are shown in Table 2.

Q5: Have you encountered instances where ChatGPT provided incorrect or misleading information about VHDL concepts or coding practices?
Q6: To what extent do you feel that relying on ChatGPT for learning VHDL may hinder the development of problem-solving skills, such as independently debugging code or researching topics?
Q7: Do you think ChatGPT's explanations and solutions can oversimplify complex VHDL topics, potentially leaving gaps in your understanding?

Table 2 Survey Result 2

Q5	**Frequently**	**Occasionally**	**Rarely**	**Never**	
	2	**10**			
Q6	**Strongly Agree**	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
	1	**5**	**5**	**1**	
Q7	**Strongly Agree**	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
	1	**4**	**5**	**2**	

This survey results indicate that most students recognized the potential downsides of misusing ChatGPT in their study of VHDL. This issue became particularly evident in the second half of the semester when the complexity of VHDL code increased. If ChatGPT-generated code failed to function as expected, diagnosing the problem proved to be exceptionally challenging.

Consequently, students who over-relied on ChatGPT early in the semester began to face challenges as the coursework became more advanced. For these students, ChatGPT turned into a hindrance rather than a helpful resource since it obstructed their ability to develop a deeper understanding and mastery of the subject. This pattern was evident in the final exam score distribution. There was significant bifurcation where students who depended heavily on ChatGPT scored poorly while those who used it thoughtfully as a complementary tool performed exceptionally well.

Balanced Approach with AI Tools

The balanced approach with ChatGPT in learning VHDL emphasizes combining the tool's strengths with active engagement in the learning process. Instead of relying on ChatGPT to generate entire code solutions, students can use it to clarify concepts, resolve specific syntax issues, or debug errors. This approach encourages learners to first attempt solving problems independently and then use ChatGPT as a secondary resource to validate ideas or refine their understanding. By integrating ChatGPT into their study process without becoming overly dependent on it students can enhance their comprehension of VHDL while maintaining critical thinking and problem-solving skills. This balanced strategy ensures that ChatGPT complements, rather than replaces, the hands-on learning necessary for mastering complex hardware description languages.

Encouraging the use of AI tools too early in the curriculum can be detrimental to student learning. The seduction is too strong for students to refrain from using it as a crutch. For example, Fig. 1 shows the VHDL code for a 1-bit full adder generated by ChatGPT. In addition to generating the code, the comments were also generated. For simple VHDL code like this, students should not need AI for the initial design. Rather, it should be used only as a debugging tool.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity Full_Adder is
5       Port (
6           A : in STD_LOGIC;      -- First input bit
7           B : in STD_LOGIC;      -- Second input bit
8           Cin : in STD_LOGIC;    -- Carry input
9           Sum : out STD_LOGIC;   -- Sum output
10          Cout : out STD_LOGIC   -- Carry output
11      );
12  end Full_Adder;
13
14  architecture Behavioral of Full_Adder is
15  begin
16      Sum <= A XOR B XOR Cin;    -- Sum calculation
17      Cout <= (A AND B) OR (A AND Cin) OR (B AND Cin);
18                                 -- Carry out calculation
19  end Behavioral;
```

Fig. 1. VHDL code of 1-bit full adder generated by ChatGPT.

Fig. 2 presents the VHDL code for a 4-bit full adder generated by ChatGPT. This code utilizes a 1-bit full adder as a component and connects four instances of it in a structured manner. Its process is depicted in the circuit diagram. Historically, many students have found structural

VHDL code challenging due to its complexity. While ChatGPT can generate such code, manual revision is recommended to ensure accuracy and adaptability. To a student without a firm understanding of VHDL design, this code may appear to be correct. However, there are several issues with it. First, the Cin in the bit-0 adder should either be grounded or connected to an input pin. In other words, the expression "Cin => Cin" should be revised to Cin => '0'. Second, the input/output names are the same for both the 1-bit and 4-bit adders, which can be confusing. To better distinguish between the inputs, the inputs in the 4-bit adder could be renamed as A_4, B_4, etc. to show that they contain 4-bits.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Four_Bit_Adder is
    Port (
         -- First 4-bit input
        A : in STD_LOGIC_VECTOR(3 downto 0);
        -- Second 4-bit input
        B : in STD_LOGIC_VECTOR(3 downto 0);
        -- Carry input
        Cin : in STD_LOGIC;
        -- 4-bit Sum output
        Sum : out STD_LOGIC_VECTOR(3 downto 0);
        -- Carry output
        Cout : out STD_LOGIC
    );
end Four_Bit_Adder;

architecture Structural of Four_Bit_Adder is
    component Full_Adder
        Port (
            A : in STD_LOGIC;
            B : in STD_LOGIC;
            Cin : in STD_LOGIC;
            Sum : out STD_LOGIC;
            Cout : out STD_LOGIC
        );
    end component;

    -- Internal carry signals
    signal C : STD_LOGIC_VECTOR(3 downto 0);

begin
    -- Instantiate the first full adder
    FA0: Full_Adder
        Port map (
            A => A(0),
            B => B(0),
            Cin => Cin,
            Sum => Sum(0),
            Cout => C(0)
        );

    -- Instantiate the second full adder
    FA1: Full_Adder
        Port map (
            A => A(1),
            B => B(1),
            Cin => C(0),
            Sum => Sum(1),
            Cout => C(1)
        );

    -- Instantiate the third full adder
    FA2: Full_Adder
        Port map (
            A => A(2),
            B => B(2),
            Cin => C(1),
            Sum => Sum(2),
            Cout => C(2)
        );

    -- Instantiate the fourth full adder
    FA3: Full_Adder
        Port map (
            A => A(3),
            B => B(3),
            Cin => C(2),
            Sum => Sum(3),
            Cout => Cout
        );

end Structural;
```

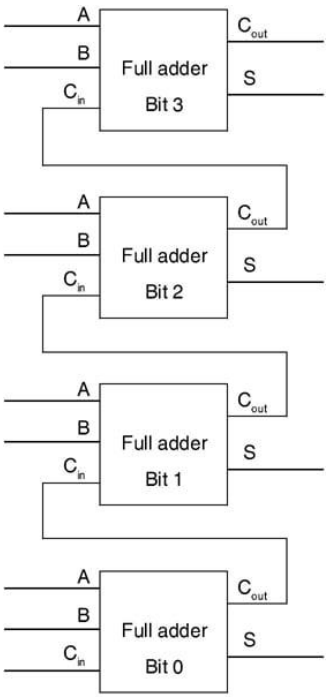

Fig. 2. VHDL code of 4-bit full adder generated by ChatGPT.

In addition to generating the VHDL code for the circuits, ChatGPT can also create the corresponding testbench file as demonstrated in Fig. 3. Typically, writing testbenches is a time-consuming and error-prone task since it requires careful consideration of various test cases,

signal initialization, and simulation conditions. This process can often lead to mistakes when handling complex designs. However, ChatGPT can significantly alleviate this burden by automatically generating the testbench code. This can ensure accuracy and save valuable time for the designers. With ChatGPT's assistance, users can focus more on the core aspects of circuit design while leaving the repetitive and meticulous task of testbench creation to the AI.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Four_Bit_Adder_TB is
-- Testbench does not have ports
end Four_Bit_Adder_TB;

architecture Behavioral of Four_Bit_Adder_TB is

    -- Component Declaration for the Unit Under Test (UUT)
    component Four_Bit_Adder
        Port (
            A : in STD_LOGIC_VECTOR(3 downto 0);
            B : in STD_LOGIC_VECTOR(3 downto 0);
            Cin : in STD_LOGIC;
            Sum : out STD_LOGIC_VECTOR(3 downto 0);
            Cout : out STD_LOGIC
        );
    end component;

    -- Signals for inputs to the UUT
    signal A : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal B : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal Cin : STD_LOGIC := '0';

    -- Signals for outputs from the UUT
    signal Sum : STD_LOGIC_VECTOR(3 downto 0);
    signal Cout : STD_LOGIC;

begin

    -- Instantiate the Unit Under Test (UUT)
    UUT: Four_Bit_Adder
        Port map (
            A => A,
            B => B,
            Cin => Cin,
            Sum => Sum,
            Cout => Cout
        );

    -- Test Process
    stim_proc: process
    begin
        -- Test Case 1: 0 + 0 + 0
        A <= "0000"; B <= "0000"; Cin <= '0';
        wait for 10 ns;
        assert (Sum = "0000" and Cout = '0') report "Test Case 1 Failed" severity error;

        -- Test Case 2: 1 + 1 + 0
        A <= "0001"; B <= "0001"; Cin <= '0';
        wait for 10 ns;
        assert (Sum = "0010" and Cout = '0') report "Test Case 2 Failed" severity error;

        -- Test Case 3: 15 + 1 + 0
        A <= "1111"; B <= "0001"; Cin <= '0';
        wait for 10 ns;
        assert (Sum = "0000" and Cout = '1') report "Test Case 3 Failed" severity error;

        -- Test Case 4: 8 + 7 + 1
        A <= "1000"; B <= "0111"; Cin <= '1';
        wait for 10 ns;
        assert (Sum = "0000" and Cout = '1') report "Test Case 4 Failed" severity error;

        -- Test Case 5: 5 + 10 + 1
        A <= "0101"; B <= "1010"; Cin <= '1';
        wait for 10 ns;
        assert (Sum = "0000" and Cout = '1') report "Test Case 5 Failed" severity error;

        -- Test Case 6: Max values
        A <= "1111"; B <= "1111"; Cin <= '1';
        wait for 10 ns;
        assert (Sum = "1111" and Cout = '1') report "Test Case 6 Failed" severity error;

        -- End of simulation
        wait;
    end process;

end Behavioral;
```

Fig. 3. Testbench file of the 4-bit full adder.

Discussion

Learning VHDL with the aid of ChatGPT offers several advantages, but it also presents trade-offs. One of the main benefits is access to an interactive and readily available resource for understanding complex concepts, debugging code, and exploring various use cases. ChatGPT can help demystify abstract ideas, such as the synthesis of hardware from code or the nuances of signal assignments versus variable assignments in VHDL. Additionally, it can allow learners to ask specific questions and receive tailored explanations. This represents a significant benefit and time saving to the learner compared to traditional methods like textbooks or forums. AI can be particularly valuable for those without access to experienced mentors or structured courses. ChatGPT can also simulate the iterative nature of problem-solving and encourage learners to refine their understanding through repeated interactions.

However, relying on ChatGPT for learning VHDL also has limitations. The AI may occasionally provide incorrect or incomplete information, especially for nuanced or domain-specific questions. Moreover, ChatGPT lacks the ability to understand and debug hardware designs in a live environment since it does not currently interface directly with simulation tools like ModelSim or Quartus. This limits its usefulness in addressing practical issues that arise during synthesis or simulation. Furthermore, overreliance on an AI assistant might discourage learners from developing critical problem-solving skills or exploring the language's official documentation in depth.

Since AI becomes a standard tool for engineers, it is essential to provide training on its appropriate and effective use. To maximize learning, it's important to complement ChatGPT's assistance with hands-on practice and a critical evaluation of its responses. Early introductions to ChatGPT in a digital systems design course should be carefully structured. Initially, students will be required to create a design without AI assistance. They will then input their design into AI to receive feedback. More extensive use of ChatGPT can be introduced in later stages of the course. In the future, new surveys will be developed to explore various aspects of AI, including its role in design, debugging, and testbench development.

Summary

AI tools can be highly beneficial for learning programming languages, including hardware description languages (HDLs) such as VHDL and Verilog. Beginners often experience frustration with syntax errors since compiler error messages frequently fail to pinpoint the exact issues. In such cases, AI tools can often more effectively identify and correct these errors. This smooths the learning curve and makes the learning process more enjoyable. Additionally, advanced HDL codes often involve numerous components where errors commonly occur at their interfaces. AI tools can assist in diagnosing and resolving these interface-related issues further enhancing the overall learning experience. Moreover, AI tools can handle tedious tasks, such as generating testbench files. This allows designers to focus on more critical aspects of their work.

An AI tool, such as ChatGPT, can be a double-edged sword for students in programming courses. On one hand it provides a valuable resource for clarifying concepts, debugging code, and accelerating problem-solving. On the other hand its ease of use and instant feedback can make it highly addictive. This can lead some students to over-rely on it. When students

habitually depend on AI to solve problems or generate code without fully engaging in the underlying logic or concepts, it can prevent their learning to reach a more advanced level.

For instructors, integrating AI tools into programming courses presents a unique challenge. They must balance encouraging effective use of these tools with ensuring students develop foundational skills through active engagement. Thoughtful course design and active mentorship are essential to managing AI's impact. This helps to ensure that it is a tool that enhances rather than diminishes the learning experience. Moving forward, we will continue exploring new approaches on applying AI tools in our courses.

References

[1] S. Shailendra, R. Kadel and A. Sharma, "Framework for Adoption of Generative Artificial Intelligence (GenAI) in Education," *IEEE Transactions on Education*, 67(5), pp. 777-785, (2024).

[2] T. Adiguzel, M. H. Kaya, F. K. Cansu, "Revolutionizing education with AI: Exploring the transformative potential of ChatGPT," *Contemporary Educational Technology*, 15(3), ep429, (2023).

[3] S. M. Vidalis, R. Subramanian. "Impact of AI Tools on Engineering Education," *Fall 2023 ASEE Mid Atlantic Conference*, (2023).

[4] K. Xu et al., " LLM-Aided Efficient Hardware Design Automation," *arXiv:2410.18582 [eess.SY]*, (2024).

[5] P. Vijayaraghavan et al., "VHDL-Eval: A Framework for Evaluating Large Language Models in VHDL Code Generation." *2024 IEEE LLM Aided Design Workshop (LAD)*, San Jose, CA, (2024).

[6] S. Hossain, A. Gohil, and Y. Wang, "Using LLM such as ChatGPT for Designing and Implementing a RISC Processor: Execution, Challenges and Limitations," *arXiv:2401.10364 [cs.LG]*, (2024).

[7] S. Liu et al., "RTLCoder: Fully Open-Source and Efficient LLM-Assisted RTL Code Generation Technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, accepted for publication, (2024).

[8] S. Qiu, T. Benjamin, and P. Hammond, "LLM-aided explanations of EDA synthesis errors." *2024 IEEE LLM Aided Design Workshop (LAD)*, San Jose, CA, (2024).

[9] J. Bhandari et al., "LLM-Aided Testbench Generation and Bug Detection for Finite-State Machines", *arXiv:2406.17132 [cs.AR]*, (2024).