Assessing the Impact of Structured Debugging Instruction in Digital Logic Design

Callaghan Wickham, University of Virginia

Callaghan Wickham is graduated 2026 from UVA with a Bachelor's of Science in Computer Science. Her interests include computer networking and automation as well as cybersecurity.

Dr. Todd DeLong, University of Virginia

Professor DeLong serves within the Department of Electrical and Computer Engineering within the School of Engineering and Applied Science at the University of Virginia.

Dr. George Prpich, University of Virginia

George Prpich is an Associate Professor of Chemical Engineering at the University of Virginia, where his research focuses on developing methods and tools to assess troubleshooting and problem-solving skills within the engineering curriculum.

Dr. Caroline Crockett, University of Virginia

Caroline Crockett is an assistant professor at the University of Virginia in the Electrical and Computer Engineering department. She received her PhD degree from the University of Michigan in electrical engineering. Her research interests include troubleshooting and conceptual understanding.

Debugging instruction in a digital logic design course

Abstract

This paper evaluates the non-cognitive effects of a debugging exercise in an introductory-level Digital Logic Design (DLD) course in an electrical and computer engineering department. Debugging and troubleshooting are the processes of identifying and fixing faults in a computer program/system, respectively. Computer science curricula have seen a gradual increase in debugging topics; however, debugging remains a sparse topic in DLD courses. Many existing resources touch on debugging to tout its importance, but do not provide scaffolding for how to learn or teach it.

The new debugging exercise involved four main steps: (1) the instructor introduced the overall activity, (2) students completed a debugging task to attempt with little guidance, (3) students observed a debugging expert attempt the same task, and (4) students completed another debugging task on their own. Throughout this process, students completed surveys and reflections to measure the impact of the activity on four non-cognitive factors: self-efficacy, mindset, emotion, and persistence.

We found that after the debugging intervention, the percentage of students who found all errors within the activity increased from 10% to 36%. Students also reported higher levels of self-efficacy, growth mindset, positive emotion, and persistence after the debugging intervention. These results suggest that the debugging intervention had an overall positive impact.

1 Introduction

Debugging is the process of identifying and fixing faults in a computer program. It is an essential task within programming, with software developers spending up to 2 hours daily fixing bugs [1]. Recognizing the importance, debugging education has become more prevalent. In 2017, around 60-70% of software developers ages 22 to 24 reported receiving formal education on debugging, compared to 40-50% of developers ages 32 to 34 [2]. Compared to debugging, hardware troubleshooting (the process of finding and fixing a fault in a physical system) has seen a lesser emphasis in formal curricula.

Digital Logic Design (DLD) is typically a second-year course in electrical and computer engineering programs that introduces students to the basics of digital hardware. When taught using a Hardware Description Language, the course is at the intersection between debugging and hardware troubleshooting, as it involves digital hardware that crosses code and hardware. Explicit debugging exercises appear to be rare in DLD courses and texts. For example, Nelson *et al.* [3] provides a strong basis for design and applications, but has no explicit teaching on how to debug the DLD realizations. The single mention of "debugging" is to define a logic analyzer as a tool that can assist in debugging. Additionally Brown and Vranesic [4] incorporates a chapter on testing for and locating faults, but does not focus on debugging and troubleshooting techniques for digital hardware systems. This paper overviews and assesses a debugging intervention in a DLD course.

The design and assessment of the debugging intervention draw heavily from Yang *et al.* [5]. Based on their review of 43 debugging studies, [5] call for debugging interventions that focus on the system comprehension step, non-cognitive debugging skills, skill transfer, and that are offered consistently throughout the semester rather than at a single time. Our research questions are:

- How successful are students at debugging a faulty circuit in DLD before and after explicit debugging instruction?
- How does a debugging exercise impact non-cognitive debugging skills?
- How is debugging success related to non-cognitive debugging skills?

We use a pre-post quantitative design to answer these research questions.

2 Background

When debugging a faulty system, an individual effective at debugging will generally follow a systematic process consisting of a few iterative steps [5]: (1) *Analyze the system:* Gain an understanding of the system and its expected behavior. (2) *Identify faulty behavior:* Compare the expected output to the actual output to verify behavior or discover discrepancies. (3) *Locate faults:* Determine the specific point in the system that causes a discrepancy. (4) *Compose and test solutions:* Try a solution to resolve faults, test if the solution was successful, and iterate until all faults are fixed. (5) *Reflection and analysis:* Walk back through the steps taken and what was and was not effective. Simon *et al.* [6] found students often skip the system analysis step of debugging.

This debugging process requires domain knowledge, system knowledge, an understanding of how to use testing equipment, and strategic knowledge [7]. Further, four primary non-cognitive skills (self-efficacy, mindset, emotion, and persistence) impact debugging performance [5].

Self-Efficacy is how effective a student believes they are at a certain task or skill [8]. For this paper, self-efficacy refers to how effective an individual believes they are at debugging.

Growth vs. Fixed mindset: An individual with a growth mindset views knowledge as elastic and believes that their current knowledge base can grow, while an individual with a fixed mindset believes they cannot become more knowledgeable [9]. Students with a growth mindset are more adaptive to different strategies, more likely to demonstrate debugging techniques covered in class, and more productively persistent through difficult bugs [5], [9].

Emotion refers to the emotions that someone experiences while debugging. Negative emotions play an important role for students when engaging in any class activities [10]. A moderate amount of negative emotions, such as frustration and anger, can motivate students to push through a challenge and reach success. However, when negative emotions are too great, students may succumb to the pressure of the assignment and give up and/or resort to alternative methods [10].

Persistence: There are two relevant types of persistence: productive and unproductive. Unproductive persistence refers to when a student is stuck on a problem and continues to make an effort without making progress [11]. This is referred to as the "wheels-spinning" phenomenon because the student is thinking but not getting anywhere, similar to a car stuck in the snow [11]. Productive

persistence is when a student continues to work towards a solution when they may come across a difficult problem or initially try an incorrect solution.

These non-cognitive skills are closely related to one another. For example, an individual with higher self-efficacy is likely to engage in productive persistence more often than an individual with low self-efficacy [12]. Individuals with a growth mindset tend to have increased persistence as they relate closely to the idea that current situations are not fact and may be improved given enough effort [9]. When a student finds a task excessively difficult, negative emotions can discourage students and decrease their self-efficacy [5]. While these skills are intertwined, it is important to recognize them separately, as they each serve their own role in the debugging experience.

3 Methods

3.1 Class Context

DLD is an introductory undergraduate course at the University of Virginia required for the Computer and Electrical Engineering BS degrees. The course teaches fundamental digital logic principles, including: "multiplexers and demultiplexers, decoders and encoders, comparators, adders and arithmetic logic unit, registers and register files, counters and timers, and register-transfer level design" [13]. Students engage with these topics directly using the Quartus Prime [14] and ModelSim [15] tools to design logic circuits, write test files, and simulate and debug their designs. DLD does not have any prerequisite courses and is typically taken during a student's second year. Around 100 students enroll in the course each semester.

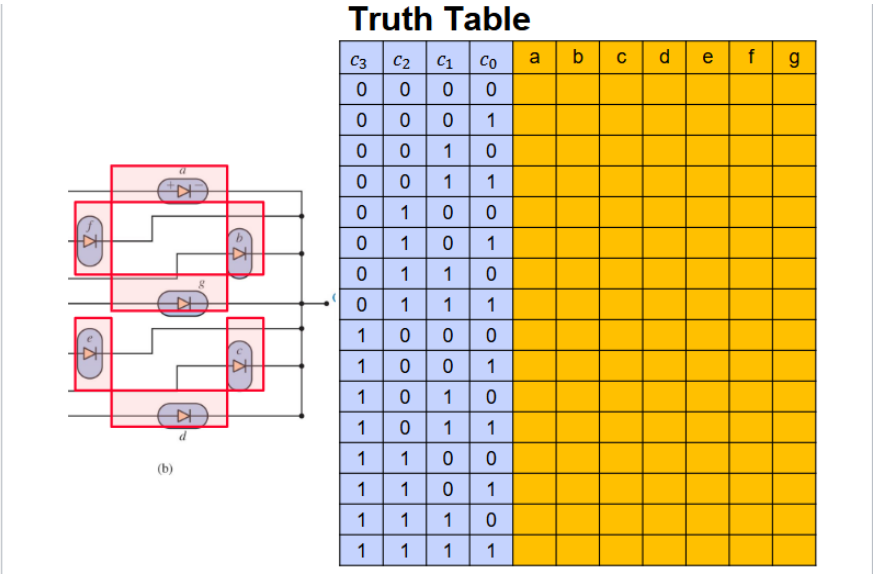
DLD consists of both lecture and in-class work days, where students are given direct support from instructors and teaching assistants (TAs) on programming assignments. Each programming assignment in DLD consists of a design and a test program. The design files define the logic for the digital hardware using VHDL. The test programs verify whether the observed outputs from the synthesized design align with the expected outputs from the system requirements.

There was no incentive for students to participate in the current research study. Of the 203 students enrolled in Fall 2025 or Spring 2026, 92 consented to the research study.

3.2 Debugging intervention

We designed the debugging intervention according to the recommendations outlined in [5]. Specifically, we encouraged the system comprehension in debugging, provided scaffolding to maximize productive persistence and minimize unproductive persistence, explicitly discussed growth mindset and the role frustration plays in debugging, and incorporated time for reflection.

The new debugging exercise in DLD was 6 weeks into the semester. The exercise built on a previous assignment to design the logic for a 7-segment LED. The lecture slides provide the system requirements and present the truth table in Fig. 1. To complete the truth table, students must indicate which LED segments are on for each possible display digit. Students must then take each column and create corresponding Karnaugh maps ("K-maps") and simplified logic expressions to control the on/off status of each LED segment based on the inputted desired display digit.



Truth Table

c_3	c_2	c_1	c_0	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Figure 1: Truth table for the 7-segment LED display problem. The first four columns represent the input: the desired display digit in binary. The remaining columns are the outputs determining whether each LED segment is on. For example, the input 1000 should display the number 8, and all LED segments should be on.

The instructor demonstrated how to fill out row 0000 and column a of the truth table in Fig. 1, how to create the K-map for output a, and how to derive the simplified logic expression from the a segment K-map. Students then derived the remaining logic expressions independently. For homework, students found the simplified logic expressions for LED segments b, c, and d; the other segments were optional practice.

In previous semesters, students then coded and tested the LED display or a similar system. To turn the activity into a debugging intervention, we introduced four new parts:

1. *Activity Introduction:* The instructor talked for roughly five minutes to explain the logistics and purpose of the activity. For the purpose of the activity, the instructor acknowledged that debugging can be frustrating but that it is an important and transferable skill that can be learned.

2. *No Guidance Debugging Activity:* Students had 20 minutes to debug a precompiled design file for the 7-segment LED display and a corresponding test file. Once time was up, students submitted their files and a summary of what errors they found. The time limit prevented excessive student frustration.

3. *Watch Video of Debugging Expert:* Students watched a 12-minute video of a TA completing the same debugging challenge.

4. *Post-Video Debugging Activity:* Students were given a new design and test file, with similar contents to part one, but with new bugs.

Students completed the first three parts during a single class session for completion credit. Students had a week to complete part 4 for homework, with their grade being majority based on completion,

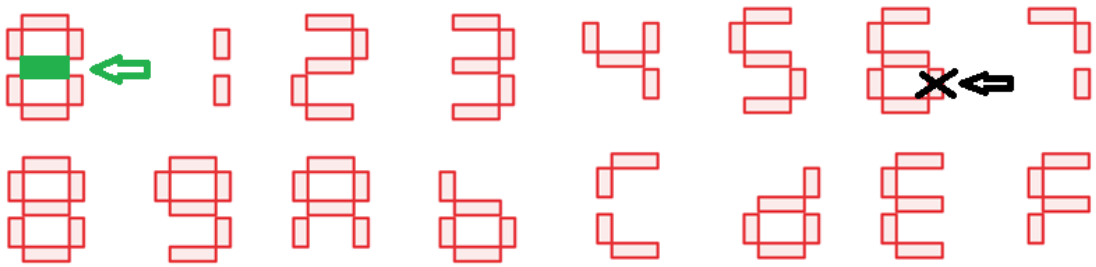


Figure 2: The sixteen LED output states representing 0 through 15 in hexadecimal. The arrows indicate the two faults from part 2: segment g is erroneously on for digit zero and segment c is missing from digit 6.

with a small percentage set aside for correctness as an incentive to find the bugs.

Students were able to access the test files throughout the debugging activities. Thus, students could read the testing strategy, modify it, and run their modified tests. Since there are 16 possible inputs, (as shown by the output states in Fig. 2), the system requires 16 total test cases (typically numbered from 0 to 15) to exhaustively test. The errors placed in the test files are: a typo in the segment declaration (sEE instead of sE), omission of test case 0, sC is declared as sc, "tabl" instead of "table" in a comment, and "verifie" instead of "verify" in another comment. The final 3 errors are inconsequential to the system, meaning they do not effect the accuracy or testing of the system.

Students had only the precompiled design file; they could not read the design file itself. Thus, students had to use observed outputs (via the wave window in Modelsim) and to reference their resources (K-maps and truth tables) to identify the specific causes of any observed errors. We designed the design bugs to encourage discrepancy detection and understanding the system as the main debugging strategies. Discrepancy detection occurs when students compare the observed output to an expected output (often in the form of a truth table) to see if the outputs align correctly. Understanding the system involves using sources such as K-maps and truth tables to understand what the system is supposed to do. Fig. 2 depicts the design errors in part 1: turning off segment c in the display of the digit 6 and turning on segment g for digit 0.[1]

The bugs in part 3 are within segments e, f, and g, which were not required K-map submissions for students. While the specifics will not be detailed in this paper, the parts 3 bugs were designed to be similar in nature and slightly more difficult to find.

3.3 Data Sources

3.3.1 Student Non-Cognitive Skill Surveys

Students took the survey in Tab. 1 before the debugging exercise began and after they completed the exercise to observe how their self-reported non-cognitive skills changed. To build this student survey, we created 2 to 3 questions per non-cognitive skill.

[1]For the simplified logic expressions, these errors correspond to removing the `not(c3) and c2` product term from segment c and the addition of `not(c2) and not(c0)` in segment g.

The *self-efficacy* questions were taken from Mamaril *et al.* [16], a study focusing on collecting data on experimental, tinkering, and design self-efficacy. All questions we based our new questions on are aimed at experimental self-efficacy, which we decided was most closely related to our debugging activity [16]. We incorporated DLD-specific information in these questions, such as ModelSim software use and verbal confirmation with peers and course staff.

We selected two questions from Karwowski [17] to gauge a student's *mindset*; one for each type of mindset. We then modified both questions to be specific to the DLD context.

When creating survey questions to measure *emotion*, we first referenced Jones *et al.* [18] and Winter *et al.* [1]. Both references found that frustration and happiness are important emotions when students are solving a design activity. We then referenced Ash and Hu [19] and Gale and Sentance [20] to create questions for these two emotions.

When creating questions to measure *persistence*, we referenced Murray *et al.* [21] and Jones *et al.* [18]. These articles stood out, as they explicitly measured student persistence whereas other surveys we considered measured factors that are related to persistence.

Skill	**Referenced Question**	**New Question**
Self-Efficacy [16]	"I can analyze data resulting from experiments."	I can analyze results using ModelSim's wave window.
	"I can orally communicate results of experiments."	I can explain my debugging process to a peer, TA, or Professor.
	"I can perform experiments independently."	I can debug DLD code independently.
Mindset [17]	"Some people are creative, others aren't—and no practice can change it"	Some people are good at debugging, others aren't — no practice can change that.
	"Practice makes perfect— perseverance and trying hard are the best ways to develop and expand one's capabilities"	Practice, perseverance, and trying hard are the best ways to develop and expand one's debugging skills.
Emotion [20]	"When I am struggling to solve an error in a program, I get frustrated."	I get frustrated when it takes me a while to resolve a bug.
	"When I solve an error, I feel happy with myself."	I feel happy when I solve a bug.
Persistence	"Setbacks don't discourage me. I don't give up easily." [21]	Setbacks within debugging don't discourage me. I don't give up easily.
	"[I] Overcome constraints in carrying out the final design." [18]	I overcome difficult bugs, even if it takes me a while.

Table 1: Non-cognitive survey.

3.3.2 Student Submissions

Following both part 2 and part 4 of the debugging exercise, students answered open-ended questions that prompted them to respond with detailed paragraphs outlining their process and what bugs they found. The prompts were:

- In the space below, write 2-3 paragraphs giving a detailed explanation of how you went about debugging. What strategies did you use? What did and didn't work well?
- Give a description of the error(s) in the provided testbench and how you resolved them. (1 paragraph per bug)
- Give a detailed description of the error(s) you believe exist in the design file and how you came to this conclusion. (1 paragraph per bug)

3.3.3 Student Reflection of Debugging Expert

Following part 3, students completed a survey to see how they received the video demonstration. This survey was a single question aimed at encouraging students to reflect on if they noticed a difference between their debugging approach and the approach of an expert. The prompt was: "How was the approach in the video similar or different to yours? Please answer in 1-2 paragraphs."

3.4 Data Analysis

A TA graded student submissions to count how many students successfully found all bugs in the design file, the test file, and in both files.

We coded pre and post-survey responses from students on the following scale: Strongly Agree (5), Agree (4), Neutral (3), Disagree (2), and Strongly Disagree (1). When presenting averages and performing significance tests, we inverted the scale for the questions on fixed mindset and negative emotion, such that larger numbers suggest higher non-cognitive skills for all questions.

4 Results

4.1 Debugging Success

Tab. 2 summarizes students' success in finding errors in the design file, the test file, and in both files. Before debugging instruction, students struggled to find the errors, with only 11% of students finding all design file errors, 36% of students finding all the test errors, and 10% of students finding all errors in both files. There was a 28% increase in success for finding design file errors from part 2 to part 4, a 17% increase in finding test errors, and a 26% increase in students who successfully found all errors in both files.

Part of debugging exercise	Design File	Test File	Both files
Part 2	11%	36%	10%
Part 4	39%	53%	36%

Table 2: Percentage of students (n=92) who identified all errors in part 1 and part 3 of the debugging exercise.

4.2 Non-Cognitive Skills Results

Fig. 3 shows the change in student-reported non-cognitive skills from part 1 to part 4. We calculated p-values using a paired two tailed t-test to evaluate if the pre and post-survey values differed significantly across the activity. The p-values were: 5×10^{-11} for self-efficacy, 0.001 for growth mindset, 0.0002 for positive emotion, and 5×10^{-6} for persistence. Based on these values, each non-cognitive skill saw a significant change ($p < 0.01$). Of the students in this study, we observed that students became more confident in themselves and more persistent after a debugging intervention. Students also reported feeling more positive emotions when debugging, with less frustration and more happiness. Growth mindset also increased among the students; however, this skill started at a high level.

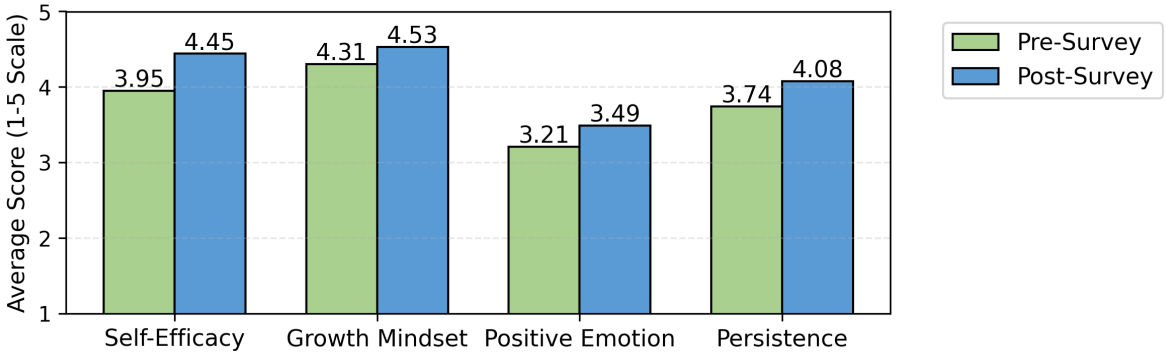


Figure 3: Student Ratings of Non-Cognitive Skills from 1 to 5 (Strongly Disagree to Strongly Agree)

Fig. 4 displays student responses to the pre and post surveys. After reverse coding the two negative questions ("Some people are good at debugging, others aren't — no practice can change that" and "I get frustrated when it takes me a while to resolve a bug."), all questions saw an increase in positive responses (agree or strongly agree). Students reported more confidence using course software, working independently, and when explaining their process to another individual in the course. They also reported an increase in a growth mindset among students as well as a decrease in frustration and increase in happiness. Students claimed to push through roadblocks more often when debugging after the activity as well.

5 Conclusion

Digital Logic and Design (DLD) is a course that integrates both hardware and software components, requiring students to engage in both software debugging and hardware troubleshooting.

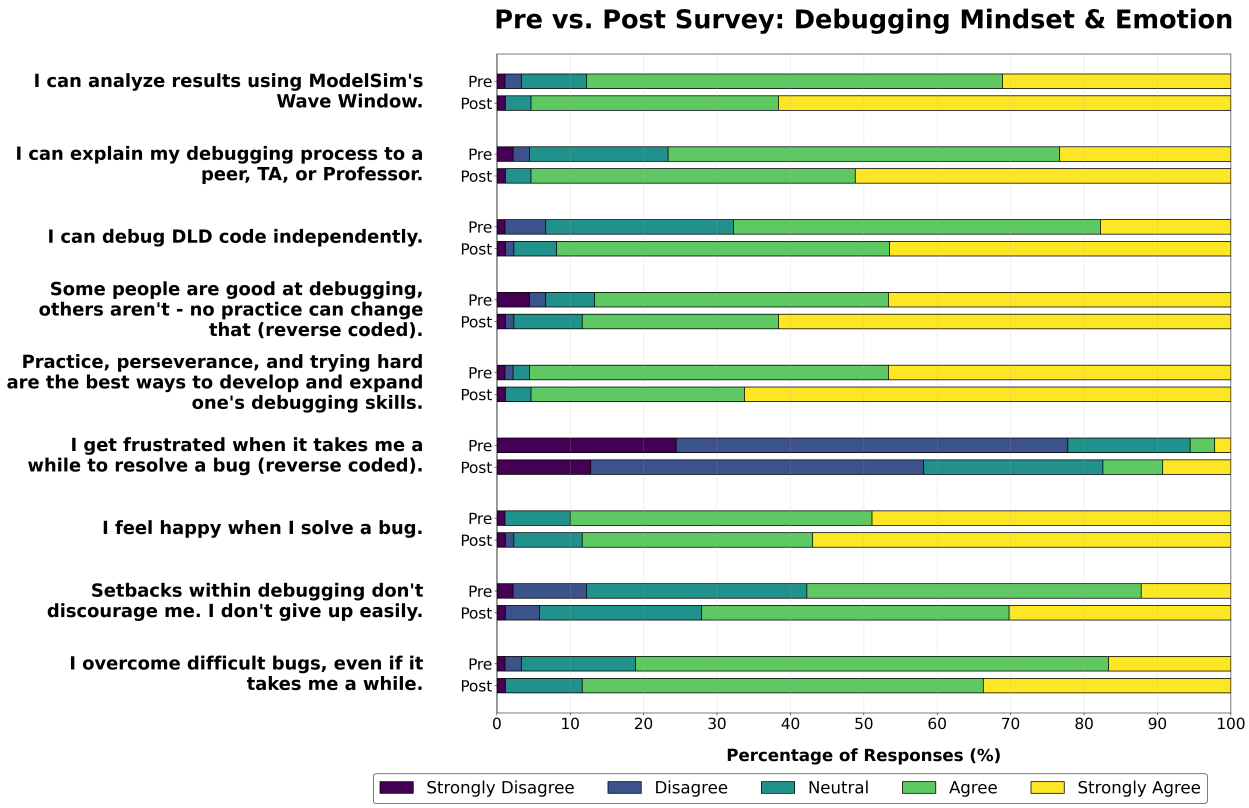


Figure 4: Student pre- and post-survey responses for non-cognitive factors. Fixed mindset and negative emotion responses were reverse coded so that strongly agree is the more positive score for all questions.

DLD courses often devote less explicit instructional time to debugging than standalone programming courses. Following recommendations from a systematic review by Yang *et al.* [5], we integrated more formal debugging instruction into a DLD course, with an emphasis on four non-cognitive factors: self-efficacy, growth mindset, positive emotion, and persistence.

Overall, we found the addition of a debugging intervention had a positive impact. There was an increase in all four non-cognitive factors among students after the activity. Students were also more successful at finding and ideating on all errors in both the design and test files, though it is worth noting that we do not know how much of the increase is attributable to the debugging exercise versus the increased amount of time to complete part 3. These results support the conclusion that intentionally embedded debugging instruction adds value to the curriculum and contributes meaningfully to students' technical skill development. Future work for digital hardware debugging research should consider if these results are transferable to other contexts.

While we structured the debugging exercise around evidence-based practices for instruction, [5] rightfully calls for debugging activities *throughout* a course. Future studies should consider expanding the instructional efforts in this paper to involve multiple debugging activities during the semester on progressively more challenging systems. A longitudinal study of such a course could then address questions such as whether a student's change in approach to debugging persists to other class exercises (or even other courses). Engineering practice naturally involves confronting errors, uncertainty, and failure, and this type of activity provides foundational training aligned

with the demands of professional engineering work. Therefore, we argue that structured troubleshooting and debugging activities should be systematically integrated throughout engineering curricula.

References

[1] E. Winter, D. Bowes, S. Counsell, *et al.*, "How do developers *really* feel about bug fixing directions for automatic program repair," en, *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1823–1841, Apr. 2023, Publisher: Institute of Electrical and Electronics Engineers (IEEE). DOI: 10.1109/tse.2022.3194188.

[2] M. Perscheid, B. Siegmund, M. Taeumel, and R. Hirschfeld, "Studying the advancement in debugging practice of professional software developers," en, *Software Quality Journal*, vol. 25, no. 1, pp. 83–110, Mar. 2017, Publisher: Springer Science and Business Media LLC. DOI: 10.1007/s11219-015-9294-2.

[3] V. P. Nelson, H. T. Nagle, B. D. Carroll, and D. Irwin, *Digital Logic Circuit Analysis and Design*, 2nd. Pearson Education (US), 2020.

[4] S. Brown and Z. G. Vranesic, *Fundamentals of digital logic with Verilog design*, en, 3. ed. New York, NY: McGraw-Hill, 2014, ISBN: 978-0-07-338054-4.

[5] S. Yang, M. Baird, E. O'Rourke, K. Brennan, and B. Schneider, "Decoding debugging instruction: A systematic literature review of debugging interventions," *ACM Transactions on Computing Education*, vol. 24, no. 4, pp. 1–44, Dec. 31, 2024. DOI: 10.1145/3690652.

[6] B. Simon, D. Bouvier, T.-Y. Chen, G. Lewandowski, R. McCartney, and K. Sanders, "Common sense computing (episode 4): Debugging," *Computer Science Education*, vol. 18, no. 2, pp. 117–133, 2008. DOI: 10.1080/08993400802114698.

[7] D. H. Jonassen and W. Hung, "Learning to troubleshoot: A new theory-based design architecture," *Educational Psychology Review*, vol. 18, no. 1, pp. 77–114, Mar. 2006. DOI: 10.1007/s10648-006-9001-8.

[8] A. Bandura, "Self-efficacy: Toward a unifying theory of behavioral change," *Psychological Review*, vol. 84, no. 2, pp. 191–215, Mar. 1977. DOI: 10.1037/0033-295x.84.2.191.

[9] C. S. Dweck, *Mindset: the new psychology of success*, en, Updated edition. New York: Random House, 2016, ISBN: 978-1-4000-6275-1.

[10] D. K. Meyer and J. C. Turner, "Re-conceptualizing Emotion and Motivation to Learn in Classroom Contexts," en, *Educational Psychology Review*, vol. 18, no. 4, pp. 377–390, Nov. 2006, Publisher: Springer Science and Business Media LLC. DOI: 10.1007/s10648-006-9032-1.

[11] H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, Eds., *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings* (Lecture Notes in Computer Science), en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-39112-5.

[12] A. Alhadabi and A. C. Karpinski, "Grit, self-efficacy, achievement orientation goals, and academic performance in University students," en, *International Journal of Adolescence*

and Youth, vol. 25, no. 1, pp. 519–535, Dec. 2020, Publisher: Informa UK Limited. DOI: 10.1080/02673843.2019.1679202.

[13] University of Virginia. "University of virginia sis campus experience: Class search (fall 2025)." Accessed via SIS Campus Experience Portal; Search parameters: Fall 2025 Catalog, Office of the University Registrar. (2025), [Online]. Available: https://sis.admin.virginia.edu/.

[14] Intel Corporation, *Intel Quartus Prime Design Software*, version 20.1.1, 2020. [Online]. Available: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html.

[15] Siemens EDA (formerly Mentor Graphics), *ModelSim Simulation and Debugging Tool*, version 20.1.1, 2020. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/modelsim/.

[16] N. A. Mamaril, E. L. Usher, C. R. Li, D. R. Economy, and M. S. Kennedy, "Measuring Undergraduate Students' Engineering Self-Efficacy: A Validation Study," en, *Journal of Engineering Education*, vol. 105, no. 2, pp. 366–395, Apr. 2016, Publisher: Wiley. DOI: 10.1002/jee.20121.

[17] M. Karwowski, "Creative mindsets: Measurement, correlates, consequences.," en, *Psychology of Aesthetics, Creativity, and the Arts*, vol. 8, no. 1, pp. 62–70, Feb. 2014. DOI: 10.1037/a0034898.

[18] S. H. Jones, B. D. Campbell, and I. Villanueva, "An investigation of self-efficacy and topic emotions in entry-level engineering design learning activities," *International Journal of Engineering Education*, vol. 35, no. 1, pp. 15–24, 2019.

[19] A. Ash and J. Hu, *WIP: Exploring the Value of a Debugging Cheat Sheet and Mini Lecture in Improving Undergraduate Debugging Skills and Mindset*, en, Jun. 2025. DOI: 10.48550/arXiv.2506.11339.

[20] L. Gale and S. Sentance, "Investigating the Attitudes and Emotions of K-12 Students Towards Debugging," en, in *The United Kingdom and Ireland Computing Education Research (UKICER) conference*, Swansea Wales Uk: ACM, Sep. 2023, pp. 1–7. DOI: 10.1145/3610969.3611120.

[21] J. M. Murray, S. Komissarov, S. L. Cook, and B. L. Murray, "Predictors for growth mindset and sense of belonging in college students," en, *Journal of Pedagogical Sociology and Psychology*, Jun. 2022, Publisher: Journal of Pedagogical Research. DOI: 10.33902/jpsp.202213791.