Design-For-Debug: A Vital Aspect In Education

Prawat Nagvajara and Baris Taskin
Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA
E-mail: {prawat_nagvajara,taskin}@coe.drexel.edu

Abstract

We often assume that debugging is a skill that comes with common sense. However, we have observed that many students do not have an inherent aptitude for debugging. Hands-on projects teaching the engineering design process can become troublesome for some students who cannot complete their projects and consequently fail their courses. In this paper, we advocate the importance of teaching debugging skills throughout digital design courses, especially during the introductory courses. We present teaching techniques in developing the skills for debugging for both introductory and advanced digital design courses. These techniques include emphasis on incremental design stages, test stimuli and observation techniques, and debugging using critical (divergent and convergent) thinking.

1. Introduction

As educators in *microsystems*, we look at education much in the same way we would look at design of a technology. That is, education is a process that can be assessed for "bugs" or debugged, improved for efficiency and thoroughness, and tested via the learning experience "feedback" from students and colleagues. We argue the necessity to improve engineering design education, exemplified by digital design education discussed in this paper, where debugging should become a critical component of design education. We emphasize the utility of debugging and bug diagnosis within a typical design process, by asserting that debugging is a skill than can be taught. We believe proper teaching of debugging skills will improve student retention rates due to fewer dropouts from design courses and empower students with trained critical thinking skills.

Although debugging in its simplest sense may appear intuitive, we believe that it is actually a skill that can be taught. Towards this end, debugging needs to be emphasized through effective learning techniques. Pragmatic approach to debugging shares the same principles with the critical thinking approach within a typical design process. The formulation of effective teaching method to communicate debugging techniques to students is also based on these principles. In essence, we find the relationship between creative thinking, creative problem solving, creativity in design, and debugging to be intriguing and instrumental to our discussion. We pose the following questions: "Are creative designs easy to debug? Do the students who are creative in design also possess good debugging skills?" It is reasonable to assume that, there would be advocates to both sides of the arguments for these two questions. On one hand, conforming to design rules and practices can simplify debugging. On the other hand it may hinder innovation. In order to develop an effective teaching technique we must understand the educational and cognitive aspects of the debugging process which is tightly coupled with the design process as a whole [4–6]. We view design-for-debug concept as not only a technical design attribute but also a product of creativity.

We have noticed a pattern of frustration among students who sense that this skill is something that is expected of them as opposed to something that can be (and needs to be) taught. As such, we have constantly observed the lack of student patience in investing the time to "fix" their designs. We believe that assessing cognitive characteristics or generic influences on learning are the basis of teaching systematic debugging and diagnostic skills. Among the assessment merits we have established are attending to salient aspects of a situation, forming relationships, conceptualizing, and generalizing as well as improving creative thinking skills (including originality, fluency, flexibility, elaboration, and resistance to premature closure). We believe teaching students the ability to assess these merits will result in improved persistence and independence of the student.

2. Teaching Techniques

In this section we will discuss plausible design-for-debug techniques for introductory and advanced digital design courses. The courses are based on Hardware Description Languages (HDLs), simulation and verification tools, electronic design automation (EDA) and hardware design using Field Programmable Gate Arrays (FPGAs). The techniques presented here include how to teach debugging skills during the design stage as well as the test bench development stage of a product design cycle. We will also discuss debugging exercises we have developed to enhance debugging skills, which utilize and stimulate creative thinking.

2.1. Design Stages

We require that students develop and present the stages in their designs starting with the simplest version, continuing with improved versions of increasing complexity and finishing with a functional version. These design stages are based on modular and hierarchical design procedures, where students are taught automatically to switch to a hierarchical design mode for advanced assignments. However, a good debugging plan, which is ignored by some students and instructors, is typically vital to the project completion. In order to teach debugging, we require students to formulate their own hierarchical design and debugging plan before they start implementation. Students are asked to itemize what potential bugs they anticipate after the design stage is complete, and explain how they would "find" and "fix" such bugs. After the design stage is complete, the students go through their list of anticipated bugs and compare it with the actual design problems they encountered. Through repetition of this process, students develop a critical understanding of how to assess and plan their designs, which translates into valuable debugging skills.

2.2. Test Bench

We introduce test bench as an integral design component in introductory digital systems classes. Students learn to design and code test benches to automate the verification process. In our proposed approach, test bench design is extended to cover techniques in capturing internal signals and states of designs for debugging. By analyzing the erroneous responses on the internal signals during simulation, students can easily deduce the design errors. Students are asked to report how they used the internal signals to capture design errors, and how they fixed these errors. The *Chip Scope Pro* tool (Xilinx [1]) also provides additional understanding to this process. Although all of these techniques are observed to be effective, we especially argue that it is more instructive for students to develop "debugging hardware" in introductory-level courses. Such a hands-on-type approach constitutes an effective mechanism to expressing and demonstrating the importance of the design-for-debugging scheme.

2.3. Debugging Exercises

Students can gain debugging skills with design exercises which contain pre-designed bugs. Many textbooks on software debugging [2, 3] view the process as a "detective" work, which involves inductive and deductive reasoning. The exercises can include designs with logical errors which can be diagnosed by simulation and synthesis errors diagnosed by test data (via test bench). The latter are more challenging exercises. They require students to retrofit a test bench onto the original design.

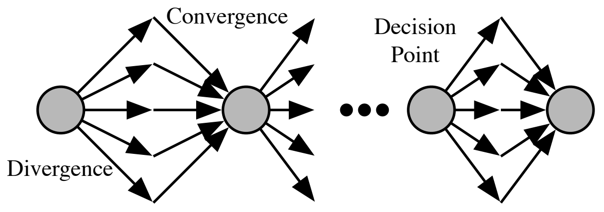

Figure 1. Divergent and convergent thinking.

Creative thinking can be a framework in developing useful debugging exercises. Figure 1 shows a diagram of creative thinking which corresponds to a strategy in debugging whereby students create different possible bugs (divergence of ideas) and converge to a bug deduced from the symptoms. The diverge/converge process continues until the design is debugged. The study of creating and solving buggy designs with the diverge/converge system enhances creative problem solving and debugging skills.

3. Concluding Remarks

Industries are putting more emphasis on developing debugging techniques to reduce the cost and the time-to-market. We argue such valuable debugging skills can be effectively taught to students through systematic exercises. We intend to improve the technical knowledge and the way students learn debugging, which will impact growth and productivity both in education and industry.

References

[1] www.xilinx.com. Xilinx Inc. website, March 2007.
[2] D. J. Agans. *Debugging: The 9 indispensable rules for finding even the most elusive software and hardware problems*. Amacom, 2002.
[3] R. C. Metzger. *Debugging by thinking*. Digital Press, 2003.
[4] National Academy of Engineering. *The engineer of 2020: Visions of engineering in the new century*. National Academy Press, Washington, D.C., 2004.
[5] R. J. Sternberg, editor. *Handbook of Creativity*. Cambridge University Press, 1999.
[6] R. J. Sternberg. Creating a vision of creativity: The first 25 years. *Psychology of Aesthetics, Creativity, and the Arts*, 1, August 2006.