WEB-BASED EXERCISES IN COMPUTER ENGINEERING[1]

Udo Hönig, Jörg Keller, Wolfram Schiffmann, FernUniversität Hagen

1 Introduction

Computer Engineering (CE) curricula ensure students' acquisition of knowledge from the discipline. In addition, students must acquire skills and working experience with the objects of the discipline. While programming tasks have become standard parts of computer science courses, integration of computer aided design (CAD) systems in computer engineering courses lags behind, partly due to the expense and complexity of these tools compared to compilers. The problem is even more difficult in distance education.

We report from our recent practise of conducting web-based exercises with CAD tools at FernUniversität, Germany's distance teaching university. The course "Computer Engineering I" introduces boolean functions, digital circuits, sequential circuits, memories, and their design. The assignments include design tasks of different digital circuits. Paper-and-pencil designs are known to be error-prone and frustrating, thus bringing motivation down. In distance teaching, there is also no face-to-face meeting with a teaching assistant (TA) where solutions are developed on the blackboard. The situation can be improved by students actually entering circuits into a CAD system. By simulation, the students can detect design errors and correct them. This increases motivation and improves the quality of submissions. Also, the students acquire skills in working with the tools of the discipline. For the TA, the advantages are improved readability of assignments and the possibility of automated correctness tests in case of many assignments. The TA can concentrate on style and quality of the designs. In a distance teaching environment, such a CAD system must be freely available, easy to distribute and easy to use. In addition, one needs a web-based system for assignment handling that integrates the CAD system. An important detail problem to be solved is self-test: students should be able to test the correctness of their design at home.

Universities with presence students have an easier job, as students are on campus. Since the 80ies, CAD systems [1] and simulators [4] installed in computer pools have been used for exercising digital design. Self-test and automated correction were not an issue. Remote access to expensive CAD systems installed at a central site has been explored in [7], remote access to lab equipment in the Emerge project, see `www.emerge-project.net`. However, the necessary long-lasting broadband interconnections are often not feasible for distance students. Moreover, the commercial CAD systems used are quite complex and difficult to use. Distance teaching courses used either text and multiple choice questions [9] or paper-based assignments [8]. The Exorciser project [11] explores web-based exercises with automated correction. However it focuses on specific tasks such as the development of a finite automaton to recognise a regular language, and uses separate java applets for each task. This leads to very high initial cost and restricts flexibility. In [2] we presented our vision of possible multimedia elements in distance teaching for computer engineering. Here we present the implementation of one of the tasks given there.

The remainder of this article is structured as follows. Chapter 2 presents the CAD system used. Chapter 3 introduces testbenches as a possibility for correctness tests. Chapter 4 presents web-based assignment handling, i.e. the integration of all tools used. Chapter 5 discusses experiences with the system in a pilot phase and evaluation of its use in a course with more than 1000 users. Chapter 6 summarises and gives an outlook on future activities.

[1]Supported by FernUniversität's Innovation fund.

2 Schematic Entry and Simulation

Exercises in computer engineering often comprise the design of logic circuits that transform binary input to output patterns. This kind of exercises poses a problem for distance learning students because of their isolation from lecturers and fellow students. Thus, they cannot discuss their solutions with others and they are unable to check the correctness of their designs in advance. In conventional paper-based exercises it often takes up to four weeks until they get back the revised and scored exercises. In order to improve the motivation and the learning situation of the students it is important to provide opportunities to test the logic circuit designs in advance. This can be accomplished by means of a CAD tool for schematics entry and simulation. Because commercial CAD tools are expensive, complex and difficult to use they cannot be used for this purpose.

HADES (HAmburger DEsign System) [6] is a portable, open source and easy-to-use CAD program that accomplishes all the needs for conducting exercises in computer engineering. By means of HADES students can enter the schematics of any logic circuit (combinatorial or sequential). In order to stimulate a circuit and to display the values of its outputs, different kinds of I/O devices are provided. In addition, hierarchical designs are supported to cope with complex designs by means of modularisation.

In contrast to commercial CAD tools HADES offers a number of other benefits:

- It can be used on any platform because it is written in Java.
- It is open source. Thus, it can be easily adapted to the specific needs of a course, e.g. the symbols of the gates can be changed to the German DIN format.
- The program size is less than 4 Megabytes. Therefore, it can be downloaded in reasonable time even via a modem connection.
- After the download, HADES can immediately be used and its usage is easy. This feature is most important for the acceptance by the students.
- While simulation in commercial CAD tools requires the start of a separate program, HADES has a built-in simulation facility. Thus, the students do not have to switch from schematic entry to simulation mode and vice versa.

3 Testbenches

Testbenches are used to check the functional correctness of a design. Students can benefit from testbenches by evaluating their solutions before submission. Later on, the same testbenches can be used for checking the submitted solutions automatically. For this purpose a so called correction server is provided. The services from this server are requested by the WebAssign platform which will be described in the next chapter. By means of the correction server the WebAssign platform informs the correctors whether the student's solution implements the required functionality or not. Therefore, the human correctors can concentrate on evaluating the design in terms of other requirements e.g. if it is in disjunctive normal form or if it conforms to other design rules such as naming conventions.

The testbenches help the students to get a quick feedback while they try to find solutions to the exercises. This improves their motivation because they have a chance to modify their solution until they find a correct one. Our evaluation of the students' performance in the assignments of the summer semester 2003 confirms this assumption (see Section 5.3).

In order to enable automatic tests we extend the logic circuit by two additional components which represent the testbench. While the first component, called the pattern generator (PG), provides stimuli signals to the logic circuit, the second component traces the response of the logic circuit to those stimuli signals. To generate the stimuli a linear feedback shift register (LFSR) is used. The LFSR has a word width of 16 to 32 bits and is initialised with a bit pattern called *seed*. Due to the feedback the LFSR produces a sequence of bit patterns that are used as stimuli signals. The response signals of the device under test are fed to a signature analyser (SA) that is realised like an LFSR. Additionally, the outputs of the test circuit are combined with the feedback in order to generate the follow-up states of the flipflops. After a given number of clock cycles a specific bit pattern will be generated in the SA. This so called *signature* depends on the properties of the PG (seed, feedback structure), the functional behavior of the device under test and the properties of the SA (seed, feedback structure). The signature summarises the function of the test circuit by a single (hexadecimal) value. It is important to notice that the signature is invariant to details of the realisation of the device under test (so called *black box test*). Thus, we can use the signature to test the functionality of a logic circuit designed by the students.

For each exercise a specific testbench is provided. To keep things simple we fix the seed and structure of the PG and the feedback structure of the SA. Only the seed of the SA is varied. In order to enable the students to self-test their designs we provide them with a seed-signature pair. Another seed-signature pair is used by the correction-server to check the submitted designs of the students[2].

4 System Integration

WebAssign is a platform that supports the interaction of the parties involved in conducting internet-based exercises [3]. One of the objectives of WebAssign is to shorten the processing times by using the internet instead of conventional mail. In the past, the exercises were printed and sent to the students via yellow mail. When the students solved the assignments they returned them to FernUniversität, from where the submissions were distributed to the correctors who are spread all over Germany and the neighbouring countries. The corrected exercises were returned to FernUniversität and finally forwarded to the students. This was complicated, time-consuming, error-prone and expensive (because of the mail charges).

WebAssign offers a web-based access to a database for the parties concerned: lecturers (or TAs), students and correctors. The lecturers can enter assignments and sample solutions for their courses. They are also able to establish deadlines for the submission of the assignments and the release of solutions, to determine the distribution of submissions among the external correctors and to access the scores of the correctors for the students' solutions. Students can access assignments and upload their solutions. When the deadline for submitting solutions is passed they also can access the sample solutions to the assignments. In accordance to the distribution given by the lecturer, WebAssign allocates the submitted solutions to the correctors. WebAssign also returns the corrected submissions to the students via email.

In order to support the correctors we extended WebAssign by a correction server that automatically checks the correctness of the submitted solutions by means of the testbenches described in Chapter 3. For each assignment the students must download a specific testbench and insert their solution. Figure 1 shows a screenshot of a testbench together with the solution module that was inserted by a student. As described in Chapter 3 the students can self-test their designs by using the testbench together with a given seed-signature pair. After they have entered the seed value in the SA, a simulation run of the complete testbench is performed. If the resulting signature in the SA coincides with the given one students know that their design solves the assignment[2]. In the same way the submitted solution is automatically checked by means of the correction server. The result of this test is stored in the database of WebAssign and forwarded to the corrector together with the student's solution.

[2] It is extremely unlikely that a faulty submission will generate the correct signature.

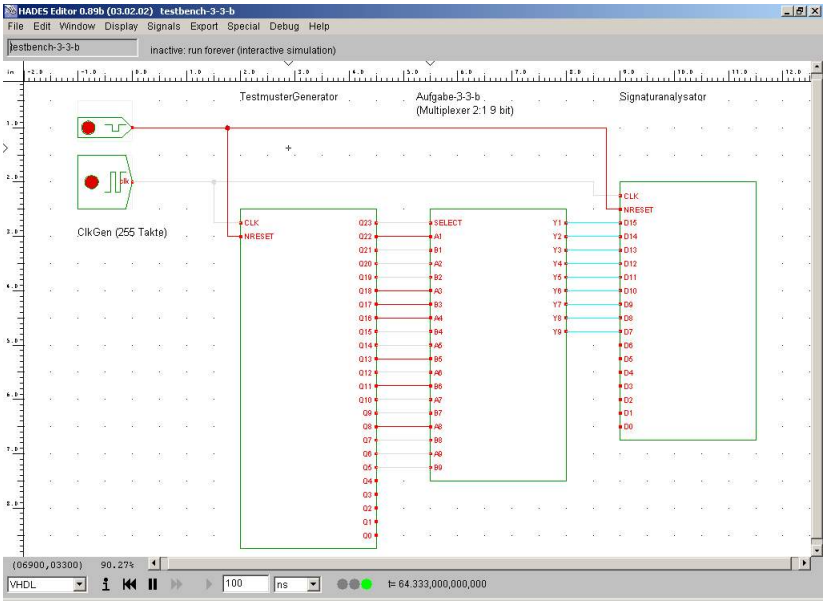

Figure 1: Screenshot of a HADES testbench with PG (left), SA (right) and the student solution (middle)

5 Results

The system described was first tested by a small group of volunteers in a pilot phase in the summer semester 2002. The experience gained encouraged a speedy introduction of the HADES/WebAssign-system into regular courses. Only minor changes were necessary before the system was ready to be used for the exercises of the course “Computer Engineering I” in summer semester 2003.

5.1 Pilot Phase

Prior to large-scale deployment in a regular course, the system’s suitability was tested with about 15 participants within a pilot phase lasting two weeks. The small number of participants and the relative short duration allowed an efficient support of the students and a fast handling of problems.

Because of the low requirements on hard- and software, the installation of HADES was easy for all pilot users. The access to the online system and the exercises functioned properly. Only the transmission of the solutions did not work well for all pilot users. These problems resulted from older browser versions at students’ computers and could be fixed by an update. Most participants characterised the handling of HADES and WebAssign as intuitive. Therefore some of the students did not use the available documentations at all. Because of the simulator features of HADES and the possibility to test their own results with provided testbenches, the students were even able to solve exercises which were more difficult than the usual ones. Finally, the participating students were asked for a review of HADES, WebAssign and the integration of both systems. The reviews underline the wide acceptance of our system by all pilot users. The number of reported problems was very small and we got only very few suggestions for improvements. Nearly all participants reported the fast correction times as the most significant advantage.

5.2 First Use in a Large Course

The course “Computer Engineering I”, a first semester course in computer engineering, consists of seven units, which are shipped to the students in a bi-weekly rhythm. To every course unit belongs an assignment which is due within two weeks. The exercises of the first two course units only consist of

multiple choice questions and questions requiring a numeric value answer. Since WebAssign is able to correct these two types of exercises from the outset, it was already used for administration and automatic correction of this course's first two assignments in summer semester 2002. In contrast, the remaining assignments include some tasks where the students design a digital circuit as solution. Up to now, WebAssign was only able to accept and distribute exercises of that kind[3] — an automatic correction was impossible.

In the summer semester 2003, the new system was deployed in a course for the first time. For this purpose, the course's assignments three to seven were transferred to WebAssign and the necessary configurations for an automatic correction were made. This work was tested in a separate test stage prior to the semester start.

The students enroled in the course were informed about the new assignment system about a month before the semester started. Only in a few exceptional cases, for example if imprisoned or handicapped, the students were allowed to submit their solutions by yellow mail. About 20 of the 1200+ actively participating students applied for this exception. From the course web site, the course's other participants could download an archive file[4] with a HADES version adapted to the course, the required design templates and the testbenches. This page also contained comprehensive user manuals of HADES and WebAssign. The majority of the participating students did not have any difficulties when downloading or installing HADES. As in the pilot phase, a few students had to install a new Java Runtime Environment or a newer version of their favourite browser. The other problems reported were all identified as faults in the students' PC configurations, e.g. wrong paths or missing files.

The majority of the students had no problems to familiarise with the new system environment. In contrast to the pilot phase, some of the participating students considered the new environment as cumbersome and difficult. Discussions with some of these students revealed that none of them had any experience with digital circuit design at all. On the other hand, the more experienced students were quite satisfied with the new system. Their criticism concentrated on some minor faults of the system's configuration, e.g. missing notifications of receipt, or broken links, which had not been found in the preceding tests. These errors were corrected as soon as they were reported.

The course ran nearly as smooth as usual. Merely the already mentioned problems with the system configuration led to an increased need for communication between students and instructors. The majority of the questions was still concerning the course contents.

One of the new system's objectives, the acceleration of the correction service, was only partially achieved. While the correction of the assignments five to seven was as quickly completed as wanted, the correction of assignments three and four lasted longer than planned. This delay can be attributed to the learning curve of the correctors, who also had to familiarise with the new system environment. It was not sufficient to give the correctors an early access to the integrated system. Instead, they should have got an opportunity to train the correction process by using old exercises or test data especially generated for this purpose.

In summary, the presented system proved its value in a large, regular course in spite of the troubles typical for every new system.

[3] The students' submissions were treated as black boxes, and were simply forwarded by WebAssign to the correctors.
[4] This file has a size of only 3,3 Megabyte.

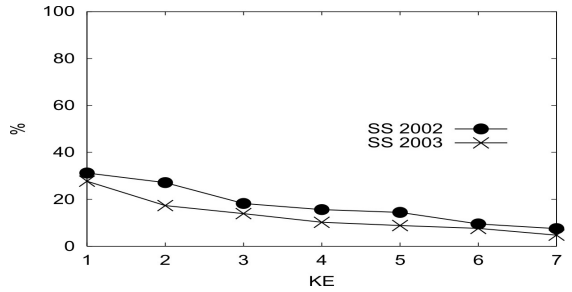

Figure 2: Number of assignments per course unit (KE), as percentage of the number of students enroled

5.3 Evaluation

We evaluated our system during its first use in a large course. We were interested in the students' acceptance and the improvement in skill acquisition and examination results. Where possible we compared with the same course from the previous year, where exercises were conventional.

To estimate acceptance of the new system, we counted the number of assignments in all seven course units (KE). Figure 2 depicts the numbers of both summer semesters (SS) as percentages of the course enrolment. We had expected in advance that initial participation would be lower, what actually became true. As both curves are quite similar apart from that initial offset, we conclude that acceptance was normal: Students willing to work with the system did not drop out at a higher rate than last year, nor did they stay longer. The initial offset might be explained by the fact that this course is also used for continuing education, where students are less inclined to take any effort to learn a new tool.

Next, we looked at course unit 3 in detail, which consists of 13 tasks. As the numbers from last year were not available, we compared tasks solved with HADES (6 in that KE) and tasks answered by text or multiple choice (7 in that KE). We found that HADES tasks are submitted by more students than were conventional tasks (411 submissions compared to 353, averaged over all tasks). Also, students achieve better results with HADES tasks according to three measures considered: First, on average only 9.35% of the students receive 0 points for HADES tasks, compared with 20.34% for conventional tasks. Second, 81.21% of the students receive the full number of points for HADES tasks, compared with 55.65% for conventional tasks. Third, students on average achieve 86.52% of the possible number of points for HADES tasks, compared to 68.28% for conventional tasks. This indicates that students using HADES know how to use it, have more motivation and show better results.

Finally, we compared the final course exams in both years. For prospects and limitations of such comparisons, see e.g. [10]. We found that more students participated in the exam (10.43% of the enroled students compared to 8.16% in 2002). Also, slightly more of the participants pass the exam (58.50% compared to 55.81% in 2002). Together, this means that more students pass the course with the help of our system.

6 Conclusions

Interactive web-based assignments are now a regular part of the course "Computer Engineering I". Experiences so far are positive, the evaluation reveals improved results in the course exam. With the help of several follow-up projects (see e.g. [5] or `www.ingmedia.de`), the system currently gets extended to the remaining CE courses of the undergraduate curriculum and to the microprocessor lab. This extension includes remote access to both simulators and real lab experiments. Thus, we hope to ameliorate distance teaching success in computer engineering.

References

1. ANDREWS D.L., THORNTON M.A. (1999) *Integration of CAD Tools and Structured Design Principles in an Undergraduate CE Curriculum,* IEEE Computer Society TC Computer Architecture Newsletter, February 1999, pp. 8–9

2. BÄHRING H., KELLER J., SCHIFFMANN W. (2001) *Use of New Media at FernUniversität Hagen (in German),* Informationstechnik und Technische Informatik, 43(4):215–218, 2001

3. BRUNSMANN J., HOMRIGHAUSEN A., SIX H.-W., VOSS J. (1999) *Assignments in a Virtual University – The WebAssign-System,* Proc. 19th World Conference on Open Learning and Distance Education, Vienna/Austria, June 1999

4. DRORDJEVIC J., MILENKOVIC A., GRBANOVIC N. (2000) *An Integrated Environment for Teaching Computer Architecture,* IEEE Micro, 20(3):66–74, May/June 2000

5. FRICKE J., SCHIFFMANN W. (2001) *Tele Lab for Digital Circuits (in German),* Informatik 2001, Part 2, pp. 1149–1153, Jahrestagung der GI/OCG, Wien, 2001

6. HENDRICH N. (1998) *HADES: The Hamburg Design System,* ASA'98, European Academic Software Award/Alt-C Conference, Oxford, September 1998

7. KAPADIA N., FIGUEIREDO R., FORTES J. (2000) *PUNCH: Web Portal For Running Tools,* IEEE Micro, 20(3):38–47, May/June 2000

8. LILJA D.J. (1999) *Education at a Distance: A Report From the Front,* Workshop on Computer Architecture Education, Orlando, FL, January 1999

9. MILENKOVIC A., NIKOLIC B., DJORDJEVIC J. (2002) *CASTLE: Computer Architecture Self-Testing and Learning System,* Workshop on Computer Architecture Education, Anchorage, AL, May 2002

10. SIX H. W., STRÖHLEIN G., VOSS J. (2001) *Evaluation of WebAssign,* 20th World Conference on Open Learning and Distance Education (ICDE Congress), Düsseldorf, April 2001

11. TSCHERTER V., LAMPRECHT R., NIEVERGELT J. (2002) *Exorciser: Automatic Generation and Interactive Grading of Exercises in the Theory of Computation,* 4th International Conference on New Educational Environments, pp. 47–50, Lugano, May 2002

Authors

Dipl.–Inf. Udo Hönig
Prof. Dr. Jörg Keller
Prof. Dr. Wolfram Schiffman
FernUniversität Hagen
FB Informatik
Postfach 940
58084 Hagen, Germany
{udo.hoenig,joerg.keller,wolfram.schiffmann}@fernuni-hagen.de