The Progressive Learning Platform for Computer Engineering

Abstract

This paper describes the Progressive Learning Platform (PLP), a system designed to facilitate computer engineering education while decreasing the overhead costs and learning curve associated with existing solutions. The PLP system is a System on a Chip design with accompanying tools reflecting a contemporary CPU architecture. It is unique in that it can be used in a number of courses (Digital Logic Design, Microcomputer Principles, Computer Architecture, Compilers, Embedded Systems) as students progress through a Computer Engineering curriculum. The system consists of a fully pipelined, MIPS like processor with surrounding support hardware. The support hardware includes a programmable interrupt controller, VGA controller and framebuffer, UART, memory controller, simple cache, timer, and GPIO hardware. All components are written in Verilog HDL, are open-source, and are freely available. To support the hardware components, a unified assembler, cycle accurate emulator, and board interface software package is included. The software is written in Java, works on Linux, Windows, and Mac OS, is open-source, and is freely available.

With only a brief learning curve on the PLP system, students can work on course objectives immediately. The system and accompanying curriculum emphasize inter- and intra-team collaborative learning by compartmentalizing components of the design process used in lab to individual teams. The goal is to expose students to a less controlled environment representative of real-world design practice. Student teams are responsible for the design decisions of their assigned component, as well as ensuring that components are compatible for use in the larger, class-wide system. Other highlights of the PLP system are: a ‘hands-on’ experience with real hardware early in the computer engineering curriculum, low overall cost for students and institutions, and cross-course application of concepts. The latter is of great importance since students often fail to see how concepts learned in one course apply to another.

With an overarching system like PLP, where different aspects of it are taught and used in a variety of courses, student can make direct connections and see how concepts in computing are related.

In this paper we present a case study of the PLP system in use in an undergraduate Computer Architecture course at Oklahoma State University. We also provide the rationale behind the development of each aspect of PLP and the expected impact on student learning, motivation, and retention.

1. Introduction

Most engineering programs use design courses extensively to give students opportunities to design, build, and test projects within realistic constraints relevant to industry practice. These courses are generally implemented with students working on team or class wide projects. These courses, especially engineering capstone design courses, are used by universities to satisfy ABET criteria [1]. All but one of the ABET outcomes can be satisfied with design courses, including:

1. An ability to apply knowledge of mathematics, science, and engineering
2. An ability to design and conduct experiments, as well as to analyze and interpret data
3. An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.
4. An ability to function on multidisciplinary teams
5. An ability to identify, formulate, and solve engineering problems
6. An understanding of professional and ethical responsibility
7. An ability to communicate effectively
8. A recognition of the need for, and an ability to engage in life-long learning
9. A knowledge of contemporary issues
10. An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

Choosing and effectively implementing a Computer Engineering design course laboratory component is a challenging task. In many Computer Engineering design courses, the complexity of designs and the ease of use of simulation tools have resulted in laboratory courses tending toward the exclusive use of small example problems, simulation and other abstractions. Students learn best from experience gained using real programs on real systems [2]. Without this, students often have difficulty relating concepts to real-world systems.

This paper introduces the Progressive Learning Platform (PLP), an FPGA based system on a chip with a complete software stack including an assembler and cycle accurate simulator, as well as an accompanying curriculum. The PLP system is intended to be used in a number of Computer Engineering courses beginning with an introductory Digital Logic Design course, through an Embedded Systems course, Computer Bases Systems, and a Computer Architecture course. The PLP system may also be used for some graduate level work. We assert that using a single, unified system throughout these courses provides an invaluable and cohesive framework that students can use to transfer knowledge and skills from one course to the next.

The PLP system and associated curriculum are based upon a particular set of teaching philosophies that include social constructivism and cooperative learning. The PLP system requires a collaborative effort by a significant number of students for design and implementation. This is facilitated by the use of a course Wiki, which is also used as a primary assessment tool, code management software, issue tracker, and special team assignments.

Although many PLP-based outcomes can be implemented and assessed individually, it is important to measure the integration of outcomes as an indicator of overall “design ability” [3]. Cheville et al. suggest that the ability to communicate the process and details of a design is a reliable measure of overall design ability [3]. This stems from the positive correlation of effective communication with performance on design projects[4] , and the use of verbal communication as a means of assessment on design projects [5] . As such, the PLP system implies communication of the design as a primary assessment tool.

Section 2 describes related work to the PLP system, both in hardware and software solutions, Section 3 describes the PLP system in technical detail, as well as its application to certain courses. Section 4 discusses how PLP is used in our Computer Architecture course. Section 5 covers an ongoing case study regarding how the PLP system impacts student learning and meeting course objectives in the Computer Architecture class. Finally, Section 6 discusses our conclusions, ongoing work, and future plans for the PLP system.

2. Related Work

Many universities use simulators to teach Computer Engineering concepts to students. Some simulators feature visual representation of the hardware to better convey the systems being studied. Examples of this include WebMIPS [6], RaVi [7] and MipsIt [8]. Other simulators such as MARS [9], SPIM [10] and TExaS [11] provide an integrated development environment and debugging features for students to develop programs for the target hardware; these systems have much less emphasis on the inner-working of the processor. Hades [12] is a Java-based logic simulator with extensive library of logic components and a powerful visualization of the circuit simulation. RaVi [7] uses Hades [12] to drive its simulation. Lastly, LC-3 [13] is an ISA with an assembler and simulator suite that students may use in learning Computer Architecture. What differentiates PLP is the tight integration between the software tool and the hardware implementation, and how PLP integrates with Computer Engineering courses. PLP is also an open project, licensed under GPLv3, and can be easily obtained from its website.

There exist several FPGA based System on a Chip designs. Holland et. al [14] suggests a reduced MIPS instruction set design for use in the classroom. The design uses an 8 instruction variant of MIPS, and allows full observability and controllability through a host tool. This design, however, does not provide a host-independent product and, thus, requires a host tool for running the processor in the classroom (for tasks that reach beyond programming). The PLP system provides a rich set of I/O, requiring the host tool for nothing more than initial programming. Additionally, the PLP MIPS design is more robust, while still simple enough for design and implementation in the classroom. Nagaonkar and Manwaring [15] discuss a complete FPGA and micro-controller based SoC for use in research and academia. Their design uses a very flexible custom hardware design. While its use in the classroom is mentioned, no explicit educational use is defined. The PLP system is specifically designed for use in the classroom, with special emphasis on exposing students to critical foundational components of Computer Engineering curriculum.

3. The Progressive Learning Platform

The core of the Progressive Learning Platform (PLP) is a complete CPU design for students to collaboratively implement in a classroom setting. The PLP system also includes a suite of software tools comprised of a programming environment and a simulator to be used along with the hardware. Hardware components of the system are written in Verilog HDL and, to maintain cross-platform compatibility, the software suite is written in Java. All PLP system components are licensed under GNU General Public License. Fritz et al. [16] provide a more detailed technical description of the PLP hardware and software environment.

3.1 Hardware

The hardware design includes the CPU, a bus arbiter, and a number of memory mapped modules (see Figure 1). The CPU is a MIPS-like machine with a 5-stage pipeline and 23 instructions. There are a number of standard modules including a UART, switches, LEDs, buttons, GPIO, timers, a VGA controller, ROM and RAMs. Additional modules can be easily created using a provided module specification. When writing new modules, a special Verilog comment is inserted above the module definition that specifies the memory map for that module. At build time, the PLP build environment reads this information and automatically creates the bus interface and memory map for all modules. Figure 2 illustrates the module definition with memory map indicator.

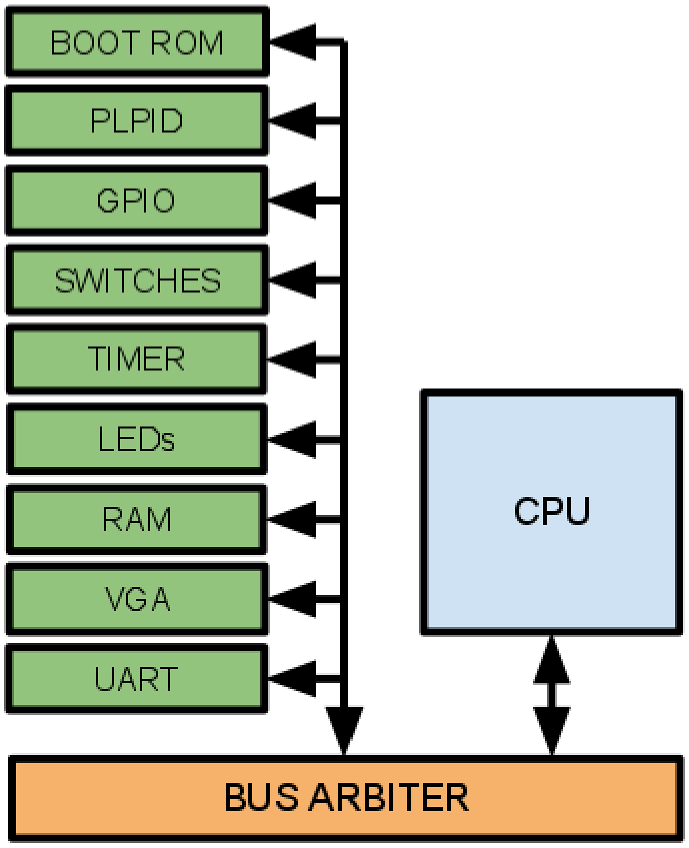


Figure 1 – The PLP system block diagram. All modules are connected via a common bus that is automatically created at build time.

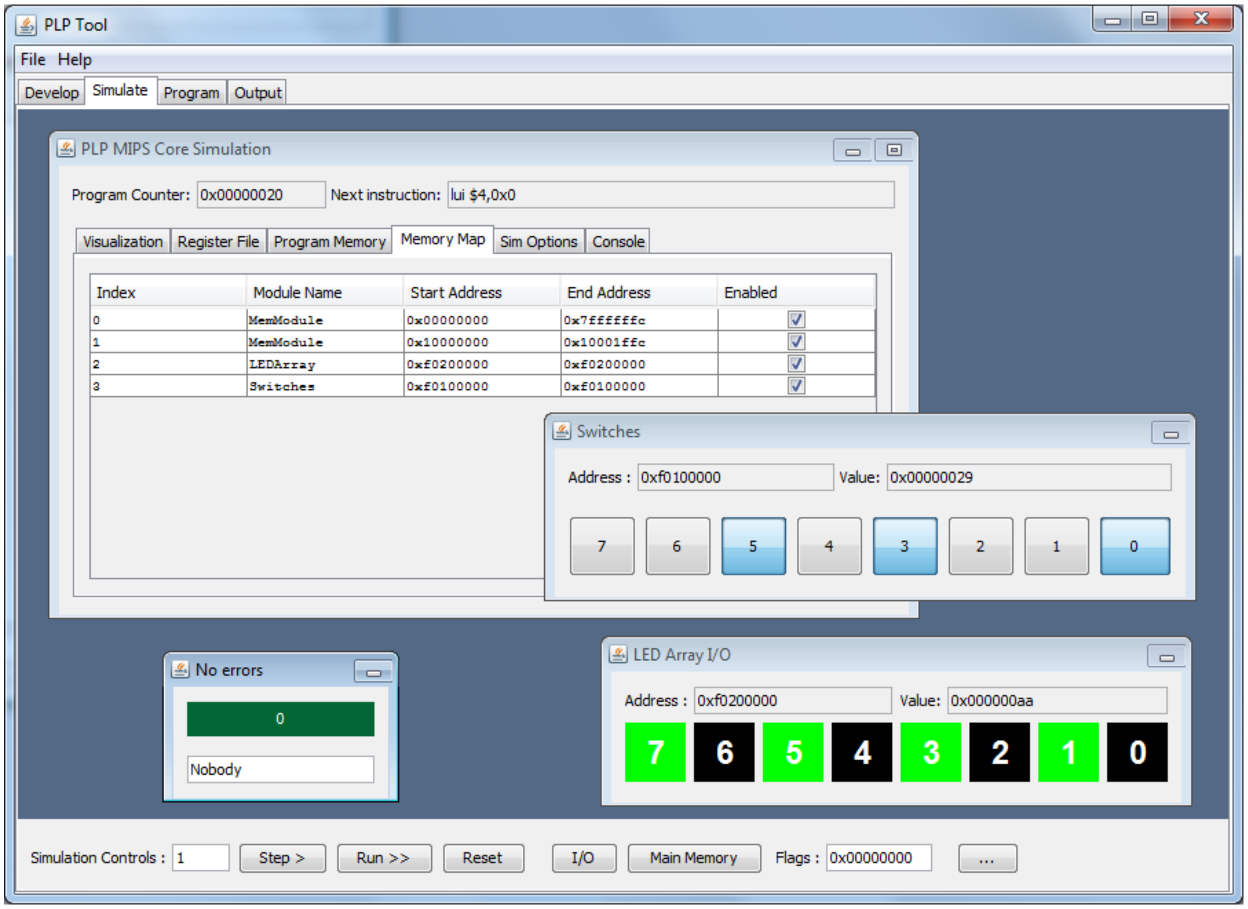
The hardware for the PLP system is described using Verilog HDL. It currently supports two development kits from Digilent Incorporated. The design is platform agnostic, however, and porting the design to another platform is relatively simple. All hardware is defined behaviorally.

```
/* PLP2MODULE start:0xf0200000 length:0x00000100 */
module my_module(rst, clk, ie, de, iaddr, daddr, drw, din, iout, dout);
```

Figure 2 – PLP module definition. A Verilog comment is used to assign the memory map, which is generated at build time.

3.2 Software

The PLP software suite is a computer architecture development and simulation framework written in Java. Users can use this framework to implement their own set of development and simulation environments by implementing Java abstract classes specified by the framework. The PLP software framework was used to create the development environment and cycle-accurate simulator for the PLP CPU design described above. Students can use the software suite to write assembly programs, simulate them, and download their programs to the board to be run. The framework also has a straightforward module interface that allows users to develop their own extensions to the simulation, such as writing an I/O module, cache simulation, etc. Figure 3 shows the simulator window with the switches and LED array I/O modules in use.



Program Counter: 0x00000020 Next instruction: lui $4,0x0

Index	Module Name	Start Address	End Address	Enabled
0	MemModule	0x00000000	0x7ffffffc	✓
1	MemModule	0x10000000	0x10001ffc	✓
2	LEDArray	0xf0200000	0xf0200000	✓
3	Switches	0xf0100000	0xf0100000	✓

Switches — Address : 0xf0100000 Value: 0x00000029

LED Array I/O — Address : 0xf0200000 Value: 0x000000aa

No errors — 0 — Nobody

Simulation Controls : 1 Flags : 0x00000000

Figure 3 – PLPTool Simulator interface. The LEDs and switches are shown with a running program.

3.3 Application to Courses

The PLP system is intended for use in a number of Computer Engineering courses from an introductory Digital Logic Design course through a Computer Based Systems and Computer Architecture Course. Additionally, the PLP system can be used in an Embedded Systems course and some graduate-level Computer Architecture courses. The use of the PLP system in a number of courses has merit in that students using the system will be able to make meaningful connections from one course to another when curriculum is based on a unified project.

3.3.1 Digital Logic Design

An introductory Digital Logic Design course is generally the first course in a series of Computer Engineering coursework. In this course, students become familiar with the FPGA and logic design principles that are foundational to the PLP system. Students also begin to use a Hardware Description Language such as Verilog (PLP hardware is implemented in Verilog). Additionally, students become proficient in testing digital logic designs, including those at the scale of the PLP system.

Students in previous iterations of this course developed a component of a MIPS SoC design, enabling them to run a pre-built software package on the system. This project was used as the final project in a series of course projects and will be extended with the use of PLP.

3.3.2 Computer Based Systems

The PLP system is well suited to teaching a Computer Based Systems course. The PLPTool provides an excellent assembler and cycle accurate simulator for students to use in small or large groups, either on a semester project or on more traditional weekly projects. We will fully incorporate the PLP system into our Computer Based Systems course in fall 2011. This will replace a long standing use of the TExaS simulator [11] for the Motorola 6800 processor.

3.3.3 Computer Architecture

The PLP system was originally intended for use in a Computer Architecture course, and this is where the current version of the system has the greatest use. Described in greater detail in the next section, students work collaboratively on a CPU design replacement for the reference design. Their CPU can be integrated into the PLP system, allowing them to run a fully featured system on a chip powered by their design.

4. How PLP is used in our Computer Architecture Classroom

4.1 Class Structure

While many engineering design laboratories are conducted in small teams, it is difficult and impractical for a small team to design and implement a complete processor in a single semester. In our undergraduate introductory Computer Architecture course, students are grouped into five large teams of five to seven students each. The teams are asked to work collaboratively on a single deliverable processor design to be used in the PLP system. The design is divided among the five teams including a front-end team, an execution engine team, a hazards and forwarding team, a test and measurement team, and a meta-team. Emphasis is placed on the highly collaborative inter- and intra-team work implied by the project assignment. To facilitate communication, the meta-team has special administrative rights over the other teams in that they are responsible for communicating inter-team communication, signal and timing definitions, and other collaborative needs. Moving the control of these components to a student team enables the instructor to further assume the role of facilitator and students to gain additional levels of real-world-applicable experience.

Each team consists of particular team roles including a team leader, documentation expert, and lead engineer. All team members must have at least one distinct role. The team leaders from each team meet regularly to ensure that proper communication of design efforts is made. The team leader for the meta-team is the primary liaison to the instructor and TAs, and he or she is effectively the project leader. In our course, the meta-team leader's role as project leader is made explicit, and he or she serves as the final decision maker on conflicts in design decisions. Again, this provides a unique opportunity to extend student learning to address the interpersonal communication challenges of teaming, a critical real-world set of skills that is too often not addressed in engineering education.

Our course emphasizes the course project and does not include any mid-term or final examinations. Weekly quizzes are used to monitor ongoing learning. Additionally, there are many in-class assignments, and teams provide weekly in-class oral status reports.

4.2 Course Project

The project, which lasts for the entire semester, is split into four phases: team-building, research, implementation, and integration. During the team-building phase in the first two weeks of the semester, students meet and exchange contact information, form basic team structure such as meeting times, and create a team behavioral contract. Students also complete required certifications for using the collaborative tools for the course, which includes a course Wiki and the code management software. Once students are certified, they are given administrative access to both tools, which allows them to modify all information, including that of other students, on the Wiki and code repository. All team-building phase information is documented on the course Wiki as the deliverable for that phase.

The research phases lasts for approximately one month and teams learn in great detail the aspects of their part of the overall design. It is during this phase that general instruction over computer architecture is provided in a lecture format. Teams are asked to learn about material relevant to their part of the design, create block diagrams, fully define signals that impact other teams, and document all of their work on the course Wiki. At the end of the research phase, teams deliver formal presentations of their findings. Other students, as well as an assessment board made up of the instructor, other knowledgeable instructors, and key graduate students, are also present for the presentation. The assessment board is responsible for assessing the team on the effectiveness and clarity of communication of their part of the design, as well as their understanding of the overall design. Other students are encouraged to ask questions as well, particularly about how that team's design impacts their own. The grade for the research phase is based on the combined perspectives of the assessment board.

The implementation phase is the longest phase of the design; within this phase, students implement their designs from the research phase in a hardware description language (in our course, using Verilog). Communication is also critical in this phase, as even moment-to-moment changes can have significant impact on the work of other teams. The meta-team is responsible for coordinating all cross-team information. All teams providing up-to-the minute documentation on the course Wiki, which enables the teams to use and build upon each other's work. The implementation phase ends with the test and measurement team evaluating the design based on the most recent version of the project specifications. This evaluation may result in teams being required to modify their designs. All implementations must meet specification in order to move on to the integration phase. Teams that have outstanding issues with their implementation do not, however, delay the overall project as the other teams may use components from the reference design to progress to the next phase.

The integration phase is the final phase of the project. In this phase, students are assembled into new teams: an integration team, a documentation team, a demonstration team, and a video team. The integration team has the task of integrating the implementations (or components from the reference design) into the final deliverable. The documentation team completes all Wiki based documentation of the design. The demonstration team uses the PLPTool to create a high-quality program to run on their design; this work will be demonstrated during the end-of-semester College of Engineering Design Day where students demonstrate their semester projects in the hallways of the Engineering college. Finally, the video team works to create a video-based documentary of the class project and student experiences with the class.

4.3 Assessment

Assessment practices in the course are based on the ability of students to effectively communicate their understanding of the design and its implementation. As noted above, this is accomplished through four major communication metrics:

1. Documentation of all work on a publicly accessible Wiki,
2. In-class demonstrations of the outcomes of each phase,
3. An end-of-term video detailing the course project, and
4. And end-of-term, high quality program for the College of Engineering Design Day.

The primary assessment metric is the course Wiki. A Wiki is a website that is driven by a powerful and simple markup language and is intended for rapid development of deeply connected content. The most prevalent example of a Wiki is Wikipedia, a free encyclopedia in which anyone can edit and contribute. Due to the rapid development of content and ease of use, Wiki software is used in numerous contexts including project development portals, documentation efforts, and in education. Wiki software facilitates collaborative development, as anyone with access to the Wiki can edit it. This allows for information to develop in an evolutionary way from multiple users. Side discussions about the development of particular Wiki articles often develop as students work to resolve conflicts of information among users. Additionally, Wiki software saves revision history of every edit to an article, allowing users to revert a particular edit to any previous point in time. Wiki software is used extensively in the engineering industry and has merit in design courses intended to be representative of industry practice.

In an educational context, Wiki software can facilitate student learning by leveraging the social constructivist and cooperative learning paradigms inherent to the collaborative nature of Wikis [17-20]. Cooperative learning in engineering design courses is an established and well received practice [21]. Students document their progress on the Wiki, allowing others to learn from the material. Consequently, students edit existing information, further facilitating learning in a social context.

Wiki software also aids in assessment since, as mentioned previously, it can measure design ability. Wiki software records individual contributions on a per user level and records every revision to existing articles. This provides an effective way to measure individual contributions as well as team contributions on a project that has one common deliverable.

Wiki software has been previously used in the classroom. Grant [22] presents a case study of Wiki usage in a secondary school. In particular, Grant investigates the use of Wiki software as a collaboration tool with emphasis on theories of community of practice. Students contributed in teams on history-based research projects. Unlike Grant's study, the PLP implementation requires that teams work together on one Wiki and collaborate on design problems that affect multiple teams.

De Pedro et al. [23] reports quantitative results from a 2-year study of Wiki software's efficacy over traditional collaborative learning techniques in the classroom. They found that in large classes (more than 15 students), Wiki software provides a clear enhancement of project quality with less overall time invested. The ability to devote less time and maintain equal or greater quality of work resulted from the replacement of traditional writing methodologies with Wiki software capabilities (such as formatting, structuring, and exchange of files). Students also reported greater satisfaction with the methodology and the resulting work.

The use of Wiki software in this course incorporates two major teaching approaches: a cooperative learning paradigm and a constructivist paradigm.

In cooperative learning, students collectively work towards a common goal that supports the learning of both the individual and the team. Cooperative learning facilitates a "positive interdependence of group members, individual accountability, face-to-face interaction, appropriate use of collaborative skills, and regular self-assessment of team functioning" [19]. The use of Wiki software facilitates cooperative learning by providing a powerful tool for rapid, asynchronous communication, discussion, revision, and scaffolding of information. The use of Wiki software also facilitates cooperative learning in a multi-generational context, as previous student's work can be retained for future semesters to build upon.

Constructivism argues that knowledge and meaning is constructed rather than pre-existing. Experiences drive the development of ideas in a continuum that the learner ultimately derives meaning from. The implementation used in the Wiki work resembles a cognitive apprenticeship approach [24], of which a critical feature is reflection. Wiki software allows students to accomplish this in an evolutionary context. Additionally, due to the collaborative nature of Wiki software, the acts of reflection support a more social constructivist paradigm [17].

5. Case Study: An Introductory Computer Architecture Course

ECEN 4243, Computer Architecture, is a senior level required course for all Computer Engineering students, and an elective for Electrical Engineering students. Class size usually is between 20 to 35 students. Student classification varies from graduating seniors to second semester juniors. The course is also open to graduate students as a pre-requisite to the two-course graduate sequence in Computer Architecture. For this particular study, participants from the Spring 2010 semester of ECEN 4243 included 33 students enrolled in the class; 18 were juniors, 14 seniors, and 1 not classified. One student of the 33 participants was female.

Research Design:

To evaluate the effectiveness of PLP a one group pretest posttest design was used. This design involves a single group that is pretested , exposed to a treatment , and then tested again [25]. The success of the treatment is determined by comparing pretest and posttest data. Specifically, we are interested in finding out if the use of the PLP system impacts students' knowledge of Computer Architecture in general, and if the use of the PLP system in the classroom impacts knowledge of concepts explicitly covered by PLP. To analyze data under this design, a dependent t-test statistical test was utilized. A dependent-samples *t* test assesses whether the mean difference between paired/matched observations is significantly different from zero. That is, the dependent-samples *t* test procedure evaluates whether there is a significant difference between the means of the two variables (test occasions or events).

Test/Instrument:

The instructor for ECEN 4243 developed the test items used in this study. They originate from a 'Quiz 0' that is administered on the first and last day of class to assess student learning. Since the test was designed to capture overall student learning over the different outcomes for ECEN 4243, some of the questions do not address topics that are directly impacted by PLP. We thus provide pre and post comparisons for the overall scores, as well as the scores for those questions on which we expected PLP to have a direct impact. The overall values reported below are for 10 questions on the quiz, whereas the PLP values reported are for 6 out of those 10 questions.

To determine the reliability (consistency) of the items, reliability analysis was performed on the data collected, which yielded a Cronbach's alpha of 0.46 for the pretest and 0.55 for the posttest. These values, though relatively low, indicate a measure of consistency on the items. Cronbach's alpha (Coefficient alpha) indicates the degree to which the items in an assessment measure similar concepts. The coefficient ranges between 0 and 1. The closer the coefficient is to 1, the higher is the reliability (higher indication of homogeneity of items).

Results:

Both overall and PLP scores for the pretest and posttest, including the sample size (N) and standard error of the mean, are displayed in Table 1.

Table 1: Descriptive Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Pre Overall	5.0400	25	1.51327	.30265
	Post Overall	6.8800	25	1.66633	.33327
Pair 2	Pre PLP	1.7200	25	1.13725	.22745
	Post PLP	3.4000	25	1.44338	.28868

A paired sample t-test was used to analyze the data. Table 2 shows that there were statistically significant differences between the pretest and posttest means in both the overall score, F (1, 25) = 6.17, p < .0001 and the PLP items score, F (1, 25) = 5.33, p < .001. In both cases, the posttest means were significantly higher than the pretest.

Table 2: Paired Samples t- test

	Mean difference	Std. Deviation	t	Df	Sig.(2-tailed)
Post – Pre (overall)	1.84	1.49	6.17	24	.000
Post – Pre (PLP)	1.68	1.57	5.33	24	.000

In general, results of this study indicated significant differences between pretest and posttest scores (both overall and PLP) for computer engineering students enrolled in ECEN 4243. These results indicate that the use of PLP intervention had a positive impact on student learning of concepts taught in ECEN 4243. These results should be interpreted with caution since participants in this study were not randomly selected and there was no control group. However, the study provides a foundation for further research in this area.

6. Conclusions and Future Work

From the use of the PLP system in our Computer Architecture course and feedback from students and other researchers, we believe the PLP system is a useful and worthwhile addition to engineering education that warrants continued research and development. The PLP system is currently used in our Computer Architecture course, and the development roadmap is in lockstep with a Computer Based Systems course that will be offered in fall 2011. The version of the PLP system that will be released in summer 2011 (for the Computer Based Systems course) will focus on simulation visualization and a debug framework. Additionally, we are incorporating elements of the PLP system in our introductory Digital Logic Design course.

We are currently conducting a three semester case study that tracks the performances and experiences of students in our Digital Logic Design course followed by the Computer Based Systems course and Computer Architecture course, all using the PLP system. This study will incorporate a number of feedback elements including student interviews and pre- and post-semester evaluations of student proficiency in course content.

7. Obtaining the Progressive Learning Platform

The PLP system is licensed under the GNU GPL version 3 license. Media components, including recorded lectures from the classroom, lecture slides, in-class assignments, and other documents such as syllabi are licensed under a Creative Commons Attribution license. All are cost-free to use and modify.

The project is hosted at http://plp.okstate.edu

8. References

1. Abet, *Criteria for Accrediting Engineering Programs, 2009-2010*. 2009: Baltimore, MD.
2. R.E. Bryant and D.R. O'hallaron, *Introducing Computer Systems from a Programmer's Perspective*, in *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*. 2001, ACM: Charlotte, North Carolina, United States. p. 90-94.
3. A. Cheville, C. Co, and B. Turner. *Communication as a Proxy Measure for Student Design Ability in Capstone Design Courses*. in *American Society for Engineering Education Annual Conference and Expo*. 2007. Honolulu, Hawaii.
4. A. Dong, A.W. Hill, and A.M. Agogino, *A Document Analysis Method for Characterizing Design Team Performance.* Journal of Mechanical Design, 2004. **126**(3): p. 378-385.
5. C. Atman, *Verbal Protocol Analysis as a Method to Document Engineering Student Design Processes.* 1998. **87**.
6. I. Branovic, R. Giorgi, and E. Martinelli, *Webmips: A New Web-Based Mips Simulation Environment for Computer Architecture Education*, in *Proceedings of the 2004 workshop on Computer architecture education: held in conjunction with the 31st International Symposium on Computer Architecture*. 2004, ACM: Munich, Germany. p. 19.
7. *Ravi*. Available from: http://ls12-www.cs.tu-dortmund.de/de/teaching/download/ravi/index.html.
8. M. Brorsson, *Mipsit: A Simulation and Development Environment Using Animation for Computer Architecture Education*, in *Proceedings of the 2002 workshop on Computer architecture education: Held in conjunction with the 29th International Symposium on Computer Architecture*. 2002, ACM: Anchorage, Alaska. p. 12.
9. K. Vollmar and P. Sanderson, *Mars: An Education-Oriented Mips Assembly Language Simulator.* SIGCSE Bull., 2006. **38**(1): p. 239-243.
10. J. Larus. *Spim: A Mips32 Simulator*. Available from: http://spimsimulator.sourceforge.net/.
11. J. Valvano. *Texas: Test Execute and Simulate*. Available from: http://www.ece.utexas.edu/~valvano/sim.html.
12. *Hades Interactive Simulation Framework*. Available from: http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/index.html.
13. *Lc-3 Simulator*. Available from: http://highered.mcgraw-hill.com/sites/0072467509/student_view0/lc-3_simulator.html.
14. M. Holland, J. Harris, and S. Hauck. *Harnessing Fpgas for Computer Architecture Education*. in *Microelectronic Systems Education, 2003. Proceedings. 2003 IEEE International Conference on*. 2003.
15. Y. Nagaonkar and M.L. Manwaring, *An Fpga-Based Experiment Platform for Hardware-Software Codesign and Hardware Emulation*, in *Proceedings of 2006 International Conference on Computer Design (CDES'06/ISBN #:1-60132-009-4/CSREA)*. 2006, CSREA Press: Las Vegas, Nevada. p. 169-174.
16. D. Fritz, W. Mulia, and S. Sohoni, *The Progressive Learning Platform*, in *Workshop on Computer Architecture Education*. 2011: San Antonio, TX.
17. M. Notari, *How to Use a Wiki in Education: 'Wiki Based Effective Constructive Learning'*, in *Proceedings of the 2006 international symposium on Wikis*. 2006, ACM: Odense, Denmark. p. 131-132.
18. J.T. Chao and K.R. Parker, *Wiki as a Teaching Tool.* Interdisciplinary Journal of Knowledge and Learning Objects, 2007. **3**: p. 57-72.
19. S. Schaffert, et al. *Learning with Semantic Wikis*. in *Workshop on Semantic Wikis*. 2006.

20. B. Mcmullin, *Putting the Learning Back into Learning Technology.* Emerging issues in the practice of university learning and teaching, 2006: p. 67-76.
21. A. Cheville, C. Co, and B. Turner. *Improving Team Performance in a Capstone Design Course Using the Jigsaw Technique and Electronic Peer Evaluation*. in *American Society for Engineering Education Annual Conference and Expo*. 2007. Honolulu, Hawaii.
22. L. Grant. *Using Wikis in Schools: A Case Study*. 2006 11/17/2010]; Available from: http://www.futurelab.org.uk/download/pdfs/research/disc_papers/Wikis_in_Schools.pdf.
23. X.D. Pedro, et al., *Writing Documents Collaboratively in Higher Education: Qualitative Results from a 2-Year Project Study.* Congreso internacional de Docencia Universitaria e Innovacion, 2006.
24. A. Collins, J.S. Brown, and S.E. Newman, *Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing, and Mathematic.* Knowing, Learning, and Instruction: Essays in honor of Robert Glaser, 1989: p. 453-494.
25. L.R. Gay, G.E. Mills, and P. Airasian, *Educational Research* 9ed. 2009, Upper Saddle River, New Jersey: Pearson Education.