# A Platform for Computer Engineering Education

**Dr. Sohum A Sohoni, Arizona State University, Polytechnic campus**

Dr. Sohoni is an Assistant Professor in Engineering and Computing Systems at Arizona State University's College of Technology and Innovation. Prior to joining ASU, he was an Assistant Professor at Oklahoma State University. His research interests are broadly in the areas of computer architecture and performance analysis, and in engineering and computing education. He has published in ACM SIGMETRICS, IEEE Transactions on Computers, the International Journal of Engineering Education, and Advances in Engineering Education. His research is supported through various internal and external funding agencies including the National Science Foundation. He is a popular and well-respected instructor, and has received many teaching awards including the Regents Distinguished Teaching Award in 2010 at OSU.

**Dr. Kerri S Kearney, Oklahoma State University**

Dr. Kerri Kearney is an associate professor of educational leadership at Oklahoma State University. Her professional experience is in both education and organizational consulting. She holds an M.B.A. and an Ed.D. Her research agenda focuses on the emotional impacts of human transition, other mothering, visual methodologies in qualitative research, and other organizational and educational topics.

**Dr. Rebecca L. Damron, Oklahoma State University**

# A Platform for Computer Engineering Education


## Abstract

The goal of the Progressive Learning Platform (PLP) pilot project is to design and test a platform to teach students how the underlying hardware building blocks relate to organization and architecture of microprocessors. PLP helps students link computer engineering concepts—logic design, microprocessors, computer architecture, embedded systems, compilers, operating systems, and high-level language constructs—in order to be able to construct a deeper understanding of the field of computer engineering. This deeper/richer understanding is expected to improve their knowledge retention, and their ability to assimilate new knowledge when they enter the workforce.

PLP is based on the theoretical framework of conceptual blending—how human beings synthesize new knowledge by assimilating and blending what they already know. Since some aspects of this framework are similar to constructivism, PLP-based courses utilize project-based learning, collaborative learning, and an emphasis on students being able to articulate design decisions.

To test the effectiveness of PLP, pre/post content tests in a number of courses served as quantitative data while qualitative results were obtained from linguistic analysis of student reflective essays and video transcript of students from lab sessions, and through focus group interviews. The qualitative methods allowed us to stand in the students' shoes and provided deep insights into how they acquired procedural and conceptual knowledge. Our results show that students viewed PLP-based course projects as authentic tasks representative of real engineering projects. The results also showed that the PLP environment served to push students to go back and review concepts from current and previous courses that they would need to apply in their projects.

The project has produced a number of products including tutorials for instructors and students, research papers, PLPTool Java code, the PLP instruction set, and the PLP reference implementation in Verilog. All these are available on the PLP website to facilitate easy adoption of PLP at other universities. We are looking for partners to adopt PLP in their courses. The long-term vision for PLP is to be a free and scalable platform for face-to-face and online education in computing worldwide. We are at the initial stage, where our platform is mature and tested at our own sites, but has not been adopted at other institutions.

Expected future direction for PLP includes three lines. Visualization tools that use PLP as the underlying engine aim to allow students at all levels (middle-school through practicing professionals) to understand how computers work. We are also working on creating literature,

website materials, and tutorials to facilitate PLP's adoption by other instructors. This work will help in the third line of exploration- research on impact of PLP on student learning.

## 1 Introduction

The Progressive Learning Platform (PLP)[1-3] is a novel, open, adaptable, multi-course Computer Engineering curriculum and technology platform developed with the help of funding from NSF's research initiation grant in engineering education (RIGEE) program. It is designed to improve student learning by closing gaps between courses in a Computer Engineering curriculum, and showing students the connections among the concepts and skills they learn in different courses. While the overarching framework for PLP is conceptual blending and material anchors (explained in the next section), the curricular materials provided with PLP for individual courses are based on the theoretical frameworks of constructivism, constructionism and communities of practice. PLP can be viewed as a vehicle for providing students with an active, collaborative, hands-on experience in the classroom. For instructors wishing to add these components to their existing courses or for instructors teaching any of the courses shown in Figure 1 for the first time, PLP provides ample support from course syllabi to project descriptions and tutorials for both instructors and students. Instructors using PLP have found it a transformative experience and students have shown high levels of engagement and motivation.

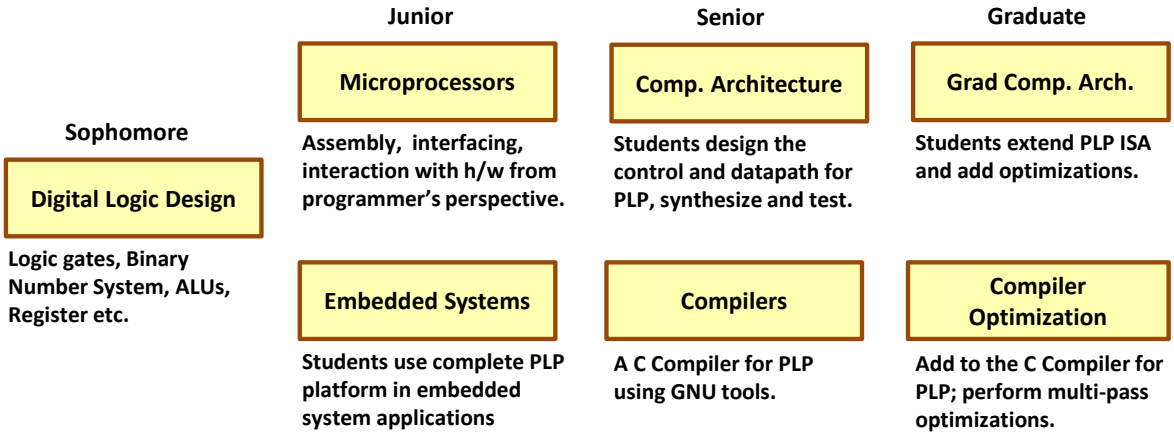


**Figure 1: Courses that can use PLP.** Ample course materials are available for a sophomore/junior course on microprocessors, and for a computer architecture course. Materials are being developed for the other courses.

On the hardware side, PLP is a System on a Chip design written in Verilog that can be synthesized on contemporary FPGA boards, with accompanying tools reflecting a contemporary CPU architecture. All hardware components of PLP 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. The PLP

hardware and software components are licensed under the General Public License version 3 to encourage open access and contribution. PLP can be downloaded free of cost from its homepage http://plp.okstate.edu. Figure 2 shows the current homepage at the time of writing this paper. A new homepage, hosted at http://plp.asu.edu is expected to replace this page at the end of the grant period (July 2013). While the current website offers direct access to the curricular elements, the mailing list, the development portal and most other useful information, user interaction design suggest a more clean and compartmentalized organization. The new website will be organized along the lines of Figure 3, to distinguish between the different aspects of PLP. The source can also be directly accessed from http://code.google.com/p/progressive-learning-platform and a public mailing list is used to serve as a communication channel between users and developers of the system.

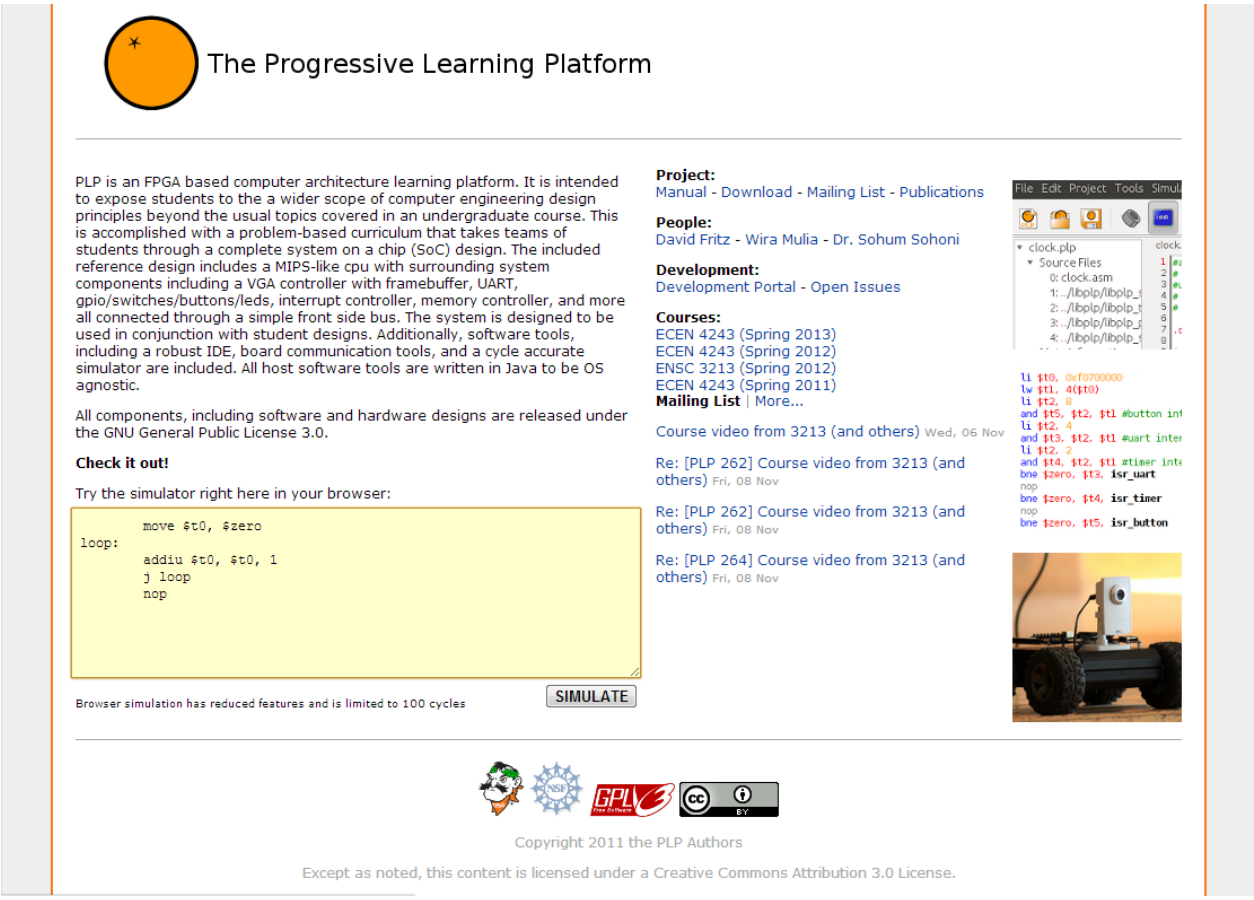


**Figure 2: PLP Homepage.** To download PLP, click the download link under Project. The mailing list is also directly accessible from the homepage, and so are the curricular elements.

This paper provides an overview of the project, with specific links for those who would like more details. The paper and the associated poster presentation are also aimed at establishing new collaborators, for conducting more robust experiments on the impact of PLP on students and instructors, and also for preventing PLP from disappearing in the valley or chasm of death[4] i.e. failing to make the transition from a research study to actual classroom adoption.

## 2 Pedagogical Foundations

***Mental Spaces, Blending and Material Anchors:*** Mental spaces[5, 6] are "conceptual packets constructed as we think and talk" ([7],p. 40). Structured by frames, which are examples of "long-term schematic knowledge", mental spaces are interconnected, and when they provide inputs to a third space that emerges from the two, this is called a blend. Not only do blends work with cognitive and cultural models, everyday things (material) such as watches, airplanes gauges and money, among others can 'anchor' blends; that is, they provide an input space in which the material and the conceptual can make joint contributions to the meaning making process. In addition, the material can provide a way of stabilizing the conceptual space[7, 8]. PLP plays a role of material anchor in conceptual blending. In other words, it acts as a trigger for students when they see it in different courses, and helps students connect the various concepts they learn in different courses. It is in this capacity, that PLP facilitates the hardware-software connection when it is used to teach hardware design, assembly programming, compilers, and operating systems.


**Figure 3: PLP's New Homepage.** This organization better reflects the different roles that PLP is expected to play: an education tool for faculty and students, an engineering education research project with some unique methods of qualitative analysis, and a development environment for hobbyists and tinkerers.

***Communities of Practice:*** In PLP, a class is set up as a kind of Community of Practice. Eckert and McConnel Ginet[9] define a community of practice as "An aggregate of people who come together around mutual engagement in an endeavor. Way of doing things, ways of talking, beliefs, values, power relations - in short, practices - emerge in the course of this mutual endeavor". The elements that make a community of practice are mutual engagement, a joint enterprise, and a shared repertoire[10]. Courses using PLP are designed, through teaming and activities, to simulate a workplace situation for students. In that way, students, working in teams, are mutually engaged in the endeavor of working with PLP. Lave and Wenger[11] see the community as a venue for learning. Learning occurs by participating in a community and therefore "questions of learning must be addressed within the developmental cycles of that community". With PLP, which simulates a workplace that reflects a community of practice, students are potentially able to learn better by interacting regularly.

***Social Constructivism and Cooperative Learning:*** The curricular materials for the Progressive Learning Platform are based social constructivist[12] and cooperative learning[13] philosophies. Social 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. This makes the student a "builder" of knowledge, as opposed to a simple recipient of knowledge. PLP applies this by acting as a material component of a "constructionist", and extension of constructivist, viewpoint[14]. Constructionism goes beyond constructivism by asserting that the best context for learning happens "when the learner is engaged in the construction of something external or at least shareable... a sand castle, a machine, a computer program, a book. This leads us to a model using a cycle of internalization of what is outside, then externalization of what is inside and so on."

PLP also applies this framework in the use of a common course Wiki for communication and assessment. The implementation used in the Wiki work resembles a cognitive apprenticeship approach[15], of which a critical feature is reflection. A number of features of PLP, especially communication as a first-class assessment metric, allow students to do this in an evolutionary way. In particular, class-wide student use of a common Wiki for documentation allows significant collaboration and reflection.

## 3 PLP Hardware

The PLP processor has an instruction set architecture (ISA) similar to the MIPS[16] processor, but with a further reduction in the ISA. Specific differences are the lack of a co-processor in PLP and no support for exceptions. Interrupts are handled in PLP through two of the 32 registers in the CPU, one for an interrupt vector and another to save the return address when an interrupt occurs. As in the pipelined MIPS micro-architecture, all hazards are forwarded except for the load/use hazard, which generates a single cycle stall.

This system utilizes Field-Programmable Gate Array (FPGA) development kits. FPGA is a reconfigurable integrated logic circuit, allowing the hardware to be configured using a hardware description language (HDL), as opposed to the “final” nature of an Application-specific Integrated Circuit (ASIC) hardware. All hardware is defined with Verilog HDL. Currently the platform targets the Digilent Nexys 2[17] and Nexys 3[18] FPGA development kits. All HDL is defined behaviorally and special structures such as block RAMs are generically defined to aid in proper inference for a number of targets. PLP is also designed to be highly portable. Porting the reference design to another target board requires only that unsupported modules be removed from the module build manifest, timing constraints be updated, and pin assignments be made. Details of the PLP hardware are available on the website[1] and in prior publications[2, 3].

## 4 PLP Software

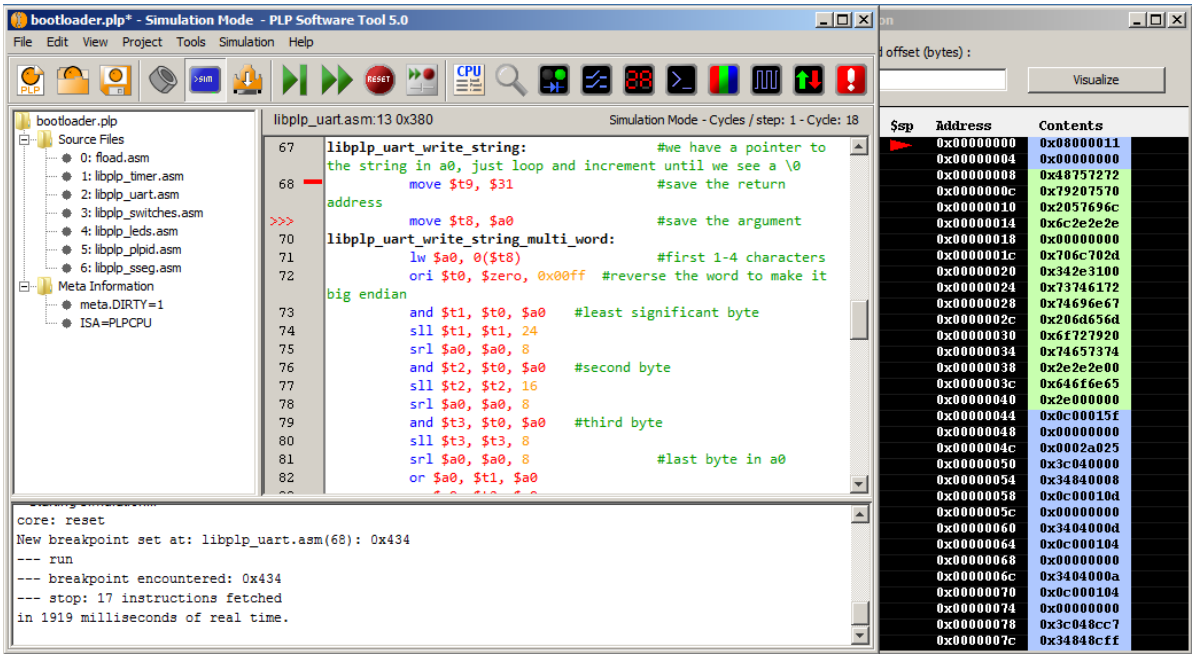


**Figure 4: PLPTool Running a Simulation of the Bootloader.** PLPTool is a self-contained editor and simulation environment that includes tools like the memory visualizer. The memory visualizer reflects the current state of the simulated memory updated in real time.

PLPTool, 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, test and debug assembly programs, simulate them to visualize their execution on the underlying hardware, and download their programs to the board. The simulator implements the I/O devices that are available on the hardware board such as LEDs, switches,

seven segments, UART, etc. The vertically integrated tool-chain includes an assembler, a cycle accurate emulator, and a board interface software package. PLPTool uses a self-contained archive as its project file, allowing easy transfer of student programs. It also provides functions to aid in understanding computing concepts and debugging such as timing visualization, breakpoints, and a watch window to monitor CPU registers, I/O registers, and the contents of the main memory. These features allow users to peek "under the hood" while their program is being executed and expose the underlying concepts that make computers tick. As an example, Figure 4 shows the main window of the software with a memory visualizer window in the background.

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 or a cache simulator. A simple cache simulator is shown in Figure 5. Such an assignment would be ideal for a graduate-level computer architecture course, especially in a software-focused degree program. By writing a cache simulator, students get exposure to underlying cache policies such as replacement of blocks, the effects of different block sizes, cache sizes etc. and at the same time utilize Java programming skills to write the simulator code. The additional advantage of using PLP, is that the simulated caches do not work in a vacuum, but are part of the PLP engine which runs real MIPS-like programs.

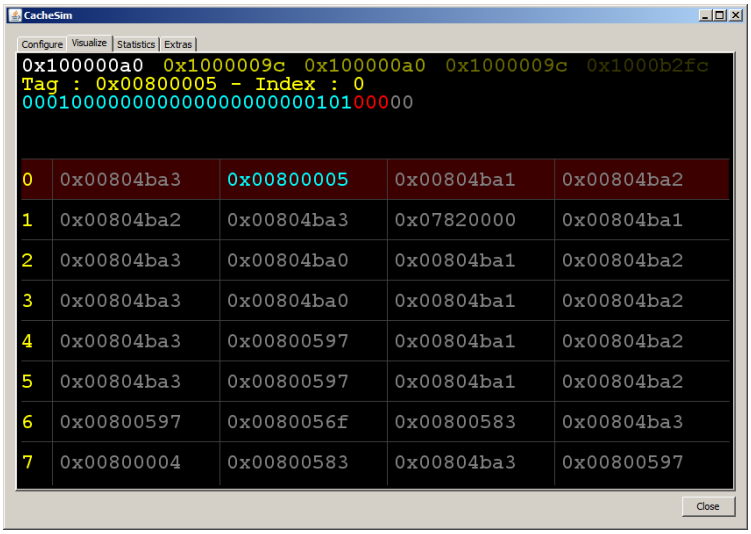


**Figure 5: The Cache Simulator Window.** The figure shows the tags of each cache line. This particular screenshot shows a cache hit in the first line of the cache.

The PLPTool IDE, and its modular Java-based design provides many exciting opportunities for faculty and students to experiment with, and for collaborative projects. As an example, one collaborator is currently using PLP as an underlying engine for visualizing the underlying computations. His goal is to create a mechanism that is separate from PLPTool's current visualization engine, one that is portable and can be replayed on a number of different devices. The modular design allows for a tracing mechanism to be developed around the PLP engine, so that a trace of all changes in the hardware can be created for later use. With an open-source

community supporting such development, this collaborator has been able to tap into the resources as well as the human know-how associated with PLP, directly communicating with the person who wrote most of the original software.

## 5 PLP Curricular Materials

PLP is designed for use in a number of Computing courses from an introductory freshman Digital Logic Design course through a senior-year Computer Architecture Course. Additionally, the PLP system can be used in Compilers, Embedded Systems, and Operating Systems, and graduate-level Computer Architecture and Compiler Optimizations courses. The use of the PLP system in a number of courses is advantageous to students' making meaningful connections from one course to another. It also eliminates the time required for mastering a new tool every semester, thereby opening up time for more content as well as more classroom interaction and projects. Depending on the environment and facilities at the host university, one can even imagine some centralized collaborative spaces for students from different courses to work together, enabling them to really see first-hand the connections between the different projects, skills and concepts. Figure 1 shows the various courses in which PLP can be used. Some of these courses are briefly described below.

**Digital Design or 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 that is later used in PLP-based courses, and with the basic logic design principles (number systems, Boolean algebra, logic gates) that can be considered the building blocks of the PLP processor. Instead of just learning what an AND gate is or what a multiplexor is, students are shown the end result (the PLP CPU) as a clear example of the application of these building blocks. 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. The application of PLP in a Digital Logic Design course also involves exposing students to a classroom environment and pedagogical style typical of a PLP-based course. Students are a part of small teams, all working towards a larger design problem, with an emphasis on communication and collaboration. It is important to implement this style correctly in this course, as this is generally the first design course student will take, and providing a poor experience can cause deeper problems, such as student departure and a distaste for team-based projects[19, 20].

**Microprocessors or Computer Organization:** PLPTool was designed specifically for a Microprocessor Organization and Programming course to allow students to write, test, debug, and step through assembly programs. PLPTool also has a detailed simulator that shows where an instruction is in the pipeline, the contents of all the CPU registers, values of memory locations, and other information pertinent to understanding how code actually executes on the hardware.

Programs written in PLPTool can also be downloaded to a PLP board, which allows the students to actually see their program control other systems with the board's collection of I/O devices instead of just being simulated on a computer. Details on the use of PLP in a microprocessors course are provided in another paper[21].

**Computer Architecture:** PLP was originally designed for and tested in a Computer Architecture course. For students who have taken a Microprocessors course that uses PLP, the PLP system serves as a material anchor that directly links the system they used to the system that they are designing. The computer architecture course introduces students to core architectural concepts, tradeoffs in various designs, and some rationales behind certain legacy and contemporary computer architectures. Throughout the course, students design a model (in Verilog) of a simple architecture and have a complete running processor by the end of class. The project extends through the entire semester, and is divided into phases. After the second and third phase (research and implementation), students present their results in class through oral presentations, and also update the course wiki. They are also asked to turn in individual reflections to capture their affective learning and self-efficacy at the end of each phase. While larger classes implement the PLP processor, smaller class sizes lend themselves to a slightly different project. In this case, students study the instruction set and are asked to add two instructions to the PLP ISA. In order to justify their additions, they have to analyze the PLP ISA and evaluate the pros and cons of their proposed instructions. They implement their changes to the PLP reference design, which requires them to study the reference implementation in detail and identify all the places where changes need to be made, as well as places that might be affected by the changes.

A prior publication[3] presents a case study describing student gains in learning, and their responses from a focus group interview in a PLP-based computer architecture course. Responses were quite positive, and the instructors observed that students were highly engaged and committed to the course project.

**Compilers:** Real-life compilers are often far too complex and production-specific to be reasonably used in the classroom as a teaching tool. Educational compilers exist in many forms, but generally implement either a toy language or just a subset of existing languages. The Progressive Learning Platform is an excellent choice for use in a compilers course as students taking the course come in with a detailed understanding of the target system and design a simple yet complete reference compiler that implements the full ANSI C language. By using PLP along with the reference PLP C compiler, we hope to offer a solution for the gap between educational "toy" compilers and full-scale production compilers. In the classroom, a theoretical discussion of the multiple phases of a compiler, compiler generation techniques, can be used, illustrated in the lab through a single semester-long course project implementing a complete, non-optimizing, C compiler for PLP. Students have access to the source of the easy to understand, well partitioned, reference PLP C compiler. Writing a full C compiler does take time, and the application of PLP

in a compilers course implies a class-wide effort, as in other courses PLP is used in, as opposed to individuals each writing their own compiler.

Additionally, advanced courses that focus on compiler optimizations instead of writing compilers can also be used with PLP. The PLP C reference compiler is a non-optimizing compiler, with specific opportunities for many compiler optimizations made easily available. Implementing compiler optimizations is impossible on complex production compilers (which already have optimizations written), and is often not worthwhile on toy compilers with limited languages.

## 6 Educational Research Methodology to Assess PLP Usage

We collected a wealth of data, both quantitative and qualitative, on student learning and collaboration. Quantitative data was obtained through a pre-post content quiz that had been in existence prior to the creation of PLP, and through a pre-post student learning mindsets survey. Three qualitative methods included focus groups conducted at the end of the semester, a set of student authored reflections collected at planned times throughout the semester, and ongoing lab observations. Peer evaluations, first using an in-house tool and then moving to CATME[22], were also used as a source of information for the classroom and for data collection purposes.

**Table 1: Data collection.** The table shows the data collection type and timing over the course of this project (crossing multiple semesters).

| Term | Course Number | Pre-Post Quiz | Student Reflections | Focus Group | Pre-Post Mindsets | Wiki | Video of Lab | Student Peer Evals |
|---|---|---|---|---|---|---|---|---|
| S 2010 | ECEN 4243 | X | | X | | | | |
| S 2011 | ECEN 4243 | X | | X | | X | | X |
| F 2011 | ENSC 3213 | X | X | | | | X | |
| S 2012 | ECEN 4243 | X | X | | | X | X | X |
| S 2012 | ENSC 3213 | X | X | X | | X | | X |
| S 2013 | ECEN 4243 | X | X | | X | X | | |
| S 2013 | CST 250 | X | | X | X | | | |
| S 2013 | CST 364 | X | X | | X | X | | X |

Quantitative and qualitative data collection techniques were designed to capture variations in participant experiences and perceptions in the cognitive and affective learning domains. Data from Spring 2013 (yet to be analyzed) will provide the first set of data from students who had a prior PLP-based course experience. As the project continues, additional data will provide insights about the impacts of PLP especially for students who use the system in multiple courses. In general, all data thus far shows positive impacts on content learning and student development of critical interpersonal skills sets.

A prior publication[3] discusses initial results from the focus group interviews for ECEN 4243. A paper presented at the 2013 ASEE annual conference[23] discusses initial results of the linguistic analysis of student reflections, and a journal article is under preparation for a detailed discussion of these results.

## 7 Summary of Findings to Date and Future Work

In summary, PLP is an open project that adapts to the needs of computer engineering education and is designed to actively engage students in the learning process. PLP was created to connect core concepts learned in various computer engineering courses, and is aimed at improving the learning experience for students. It is grounded in the theories of social constructionism and situated cognition. Results from the pilot studies show that PLP is highly effective in engaging students and in helping them gain valuable skills. One clear advantage we are beginning to see is that students, instructors, and teaching assistants all found it very convenient to use the same system (PLP) in multiple courses. PLP can be viewed as a vehicle for providing students with a classroom experience based on social constructionism and communities of practice, and instructors wishing to incorporate project-based, active and collaborative learning are highly encouraged to visit the website and contact the authors.

Results from the writing reflections show that students are able to articulate both their content knowledge and the strengths and weaknesses of their collaborative work. Reflections tended to become more detailed as the semester progressed, and thus, review of earlier reflections established individual and communal learning as well as pinpointed significant moments of learning in the semester. The written reflections showed that students were able to articulate their acquisition of procedural knowledge and comment on the value of that knowledge acquisition. Results also showed that students gained conceptual knowledge in conjunction with procedural knowledge, and applied this conceptual knowledge (often going back to previous lectures/slides) in their projects. This finding is important because PLP was intended to bridge the gap between lecture and lab and to provide a context in which students could apply their conceptual knowledge.

Further work will be done with videotaped data to identify how students talk about PLP and the PLPBOT in order to determine the role of PLP in their learning processes. Videotape transcripts will also provide insight into how students collaborate. The data we have collected is much more than what can be processed in the pilot grant’s timeframe, and we will apply for further funding to refine our experiments, collect more specific data as guided by our current findings, and further analyze the data collected through the pilot project.

One of our observations is that PLP in its current state may be a challenge for other instructors to

readily adopt it in their courses. We are addressing this by improving the documentation, standardizing and rewriting parts of the code, and beta-testing PLP at other universities. We are looking for collaborators for this venture as well, not only instructors who can test PLP in their courses, but also researchers in engineering education, who can help set up more robust experiments at these sites and help with the data analysis.

**Acknowledgment**

This work was supported in part by NSF award EEC 1136934

References

[1] *The Progressive Learning Platform Website* http://plp.okstate.edu.

[2] D. Fritz, W. Mulia and S. Sohoni, "The progressive learning platform," in *Workshop on Computer Architecture Education,* San Antonio, TX, 2011, .

[3] W. Mulia, D. Fritz, S. Sohoni, K. Kearney and M. Mwavita, "PLP: A Community Driven Open Source Platform for Computer Engineering Education," *International Journal of Engineering Education,* vol. 29, pp. 215, 2013.

[4] J. Fairweather, "Linking evidence and promising practices in science, technology, engineering, and mathematics (STEM) undergraduate education," *Board of Science Education, National Research Council, the National Academies, Washington, DC,* 2008.

[5] G. Fauconnier, *Mappings in Thought and Language.* Cambridge University Press, 1997.

[6] G. Fauconnier, *Mental Spaces: Aspects of Meaning Construction in Natural Language.* Cambridge University Press, 1994.

[7] G. Fauconnier and M. Turner, *The Way we Think: Conceptual Blending and the Mind's Hidden Complexities.* Basic Books, 2003.

[8] E. Hutchins, "Material anchors for conceptual blends," *Journal of Pragmatics,* vol. 37, pp. 1555-1577, 2005.

[9] P. Eckert and S. McConnell-Ginet, "Think Practically and Look Locally: Language and Gender as Community-Based Practice," *Annual Review of Anthropology,* vol. 21, pp. 461, 1992.

[10] E. Wenger, *Communities of Practice: Learning, Meaning, and Identity.* Cambridge University Press, 1998.

[11] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation (Learning in Doing: Social, Cognitive and Computational Perspectives).* Cambridge University Press, 1991.

[12] M. Cakir, "Constructivist Approaches to Learning in Science and Their Implications for Science Pedagogy: A Literature Review," *International Journal of Environmental & Science Education*, vol. 3, pp. 193-206, 2008.

[13] J. Shimazoe and H. Aldrich, "Group Work Can Be Gratifying: Understanding & Overcoming Resistance to Cooperative Learning," *College Teaching*, vol. 58, pp. 52-57, 2010.

[14] I. E. Harel and S. E. Papert, *Constructionism.* Ablex Publishing, 1991.

[15] L. B. Resnick, *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Lawrence Erlbaum, 1989.

[16] J. Hennessy, N. Jouppi, S. Przybylski, C. Rowen, T. Gross, F. Baskett and J. Gill, "MIPS: A microprocessor architecture," in *Proceedings of the 15th Annual Workshop on Microprogramming*, Palo Alto, California, United States, 1982, pp. 17-22.

[17] *Digilent Nexys2 Spartan-3E FPGA Board* http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS2.

[18] *Digilent Nexys3 Spartan-6 FPGA Board* http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS3.

[19] J. M. Braxton, J. F. Milem and A. S. Sullivan, "The influence of active learning on the college student departure process: Toward a revision of Tinto's theory," *Journal of Higher Education*, pp. 569-590, 2000.

[20] A. Nora, A. Cabrera, L. Serra Hagedorn and E. Pascarella, "Differential impacts of academic and social experiences on college-related behavioral outcomes across different ethnic and gender groups at four-year institutions," *Research in Higher Education*, vol. 37, pp. 427-451, 1996.

[21] S. Sohoni, D. Fritz and W. Mulia, "Transforming a microprocessors course through the progressive learning platform," in *ASEE Midwest Section*, Russelville, AR, 2011, .

[22]*CATME Smarter Teamwork* www.catme.org.

[23] R. Damron, S. Sohoni, K. Kearney and Y. Cho, "Impact of PLP on student learning: Initial results," in *ASEE Annual Conference*, Atlanta, GA, 2013, .