122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Making Value for Society

Paper ID #12860

Learning about Digital Logic by Discovery

Prof. Joanne Bechta Dugan, University of Virginia

Joanne Bechta Dugan is Professor of Electrical and Computer Engineering and the Director of the Computer Engineering Programs at the University of Virginia. Her research focuses on probabilistic assessment of the dependability of computer-based systems. She has developed the dynamic fault tree model, which extends the applicability of fault tree analysis to computer systems. Dugan holds a B.A. degree in Mathematics and Computer Science from La Salle University, and M.S. and PhD degrees in Electrical Engineering from Duke University. She is a Fellow of the IEEE.

Learning about Digital Logic by Discovery

Abstract

The creation of interactive models of digital electronic circuits allows students to learn about their design and function by active exploration and discovery. Students can explore models of circuits which are fundamentally important to the design of computing devices. Such exploration is directed by the students' curiosity; an exploration guide draws attention to important aspects. Complex systems can be broken apart into constituent pieces which can be examined separately and in concert. New designs can be implemented using constituent parts from other systems. This approach allows/encourages students to engage more deeply with the material, providing a mechanism for them to explore alternatives and create new systems. Despite markedly increasing class sizes, students report higher levels of satisfaction and enjoyment with this approach, demonstrate deeper understanding of concepts and exhibit an enhanced ability to apply these concepts in innovative solutions.

Introduction

Digital Logic Design is a standard introductory course in computer engineering taught in one form or another almost everywhere. At the author's institution, it has no prerequisites, and is required by electrical engineering, computer engineering and computer science students. DLD, as the course is locally known, has long enjoyed a reputation as a fun class, and it's been a popular choice for non-majors as an elective. In recent years, rapidly increasing class size and a move to a larger lecture hall resulted in an increased barrier between the instructor and the students; students became more passive and absenteeism increased. Hence there was a need to increase student engagement, to help overcome the barriers created by increasing class sizes, and to restore the sense of fun.

DLD has been a topic of considerable interest at ASEE for many years and recent ASEE publications reflect several trends. The prevalent use of hardware description languages (HDL) and programmable logic device (PLD) implementation enables designers to develop increasingly complex devices at a higher level of abstraction. One trending ASEE topic describes the increased incorporation of HDL and PLD into introductory classes (McCarthy, Wright, Barrett, & Hamann, 2010) (Yilmazer & Sekar, 2011) (Alaraje, Sergeyev, & DeGroat, 2011) and the subsequent concern that students may not understand fundamental concepts when starting at such a high level of abstraction (Carroll, 2012). The work presented in this paper does not relate to the use of HDL and PLD as these topics are covered in a later course at the author's institution.

A second common topic in recent ASEE publications about DLD describe the use of active and interactive learning approaches and realistic examples, such as PETL (Wang, 2010), problem-based learning (Yilmazer & Sekar, 2011), project-based learning (Carroll, Geiser, & Levine, 2014), example-based learning (Hoffbeck J. , 2014) and visualization (Devore & Soldan, 2012).

The work presented in this paper fits in this topic as it presents new interactive materials and examples that increase engagement, even in large (enrollment >150) classes.

A third group of papers recently published in ASEE were concerned with the specifics of teaching and learning about concepts in DLD. Peterson & Clark (Peterson & Clark, 2012) developed PRISM (Programmable Reconfigurable Informational Simple Microcomputer) a visual simulation of a 4-bit architecture computer with the goal of teaching how a system processes and executes instructions. PRISM simulates execution of an assembly language program (an assembler is included), highlighting the active participants and paths (according to logic values). Devore & Soldan have developed a software tool called *VisiBoole* that simulates a Boolean logic equation and displays logic values and dependencies in a spreadsheet-like approach (Devore & Soldan, 2012). The work presented in this paper fits in this context as the interactive materials allow students to directly engage with the concepts and discover functionality.

Overview and History

Three years ago a DLD student discovered a very nice (free, open-source) software package (called *Logisim* (Burch)) that enabled the development of simulated circuits for digital design; the student had used it to enhance his own learning. At the end of the semester this student shared his discovery with the instructor who was immediately impressed with the functionality and ease of use. The instructor had tried using similar software systems but none were acceptable for classroom use, either being too expensive, too buggy, too limited in functionality, too simplistic or too hard to use. *Logisim* is an excellent software program for instructional use in a college course and the instructor began using it in fall 2011 in a limited capacity, increasing usage each semester as the capabilities and correctness of the software were verified and as new materials were developed. The instructor built several circuit models in *Logisim* and these circuits were sometimes used in class and sometimes made available to students as supplementary materials. Students reported finding these example circuits helpful as an interactive simulation of a circuit is much more engaging than a flat diagram printed on paper.

Unfortunately, it was determined that *Logisim* did not adequately support the last two main topics in the class (RTL design and programmable processor design) because examples of these types of circuits implemented in *Logisim* were by necessity complex and detailed, and were also difficult to change. These last two topics were already very challenging and the difficulty seemed to be enhanced by the loss of *Logisim* as a supporting technology. Students came to depend on *Logisim* to help them understand the more complex circuits up to this point in the class, but *Logisim* failed them at this critical juncture. The solution to this problem would require a significant enhancement to the *Logisim* package.

Understanding these last two topics is difficult because they require students to understand two different types of systems (datapath and controller) that operate concurrently and influence each other. So there were two complex circuits operating at the same time; the outputs from one were the inputs to the other and vice versa. The circuits were time-dependent (i.e. contained memory) so that the behavior evolved in time as well as being interrelated.

The instructor had tried several approaches to enhance understanding of these concepts, including manual simulations using multiple colors to trace time dependencies, using physical objects (such as toy figures) that move across the page (displayed using a document camera), human dramatizations (using student volunteers to play the roles of different components) and developing complex computerized animations. Each of these techniques helped, but none were completely satisfactory. Students might understand the concepts as they were illustrated in class but they were unable to take adequate notes for later review. Animations were excellent for illustrating concepts but were very time consuming to develop and did not allow for easy alteration. Thus the coverage of these last two topics was unsatisfactory from both the student and instructors perspective.

In spring 2013 we developed a promising approach to this problem. Two undergrad students developed new functionality for *Logisim* (under the instructor's supervision) that enabled it to be used for understanding those last two topics in the course. This enhancement was not trivial but we hoped it could break through the barrier to understanding those last two very complex concepts. The new *LogisimFSM (*as we called it*)* was capable of supporting learning throughout the semester. All that remained was the development of materials using the new *LogisimFSM*; these materials could enable students to engage more fully with the concepts in a repeatable manner.

The resulting *LogisimFSM* package sparked two ideas for active learning techniques that could improve student engagement in a large class on Digital Logic Design: *guided explorations* and *karaoke design*. The development of a set of materials for both techniques was funded by a small internal grant in summer 2013. The materials were used and refined in fall 2013 (128 students), spring 2014 (156 students) and fall 2014 (144 students). The results were deeply satisfying. Not only did students engage more fully with the material (and frequently reported on having fun while doing so), their understanding of those two most complex topics was significantly enhanced. Students were able to complete more complex assignments and answer more involved test questions on these concepts than in previous semesters.

Guided Explorations

Instead of teaching *about* a component in the traditional manner (this is a decoder, here is how it's used, here's how it's typically designed), students are instead given a set of "mystery" circuits (in *Logisim*) and an "exploration guide" and are tasked with solving the mystery. The end result is the same (they learn about decoders) but they are intrigued by the mystery.

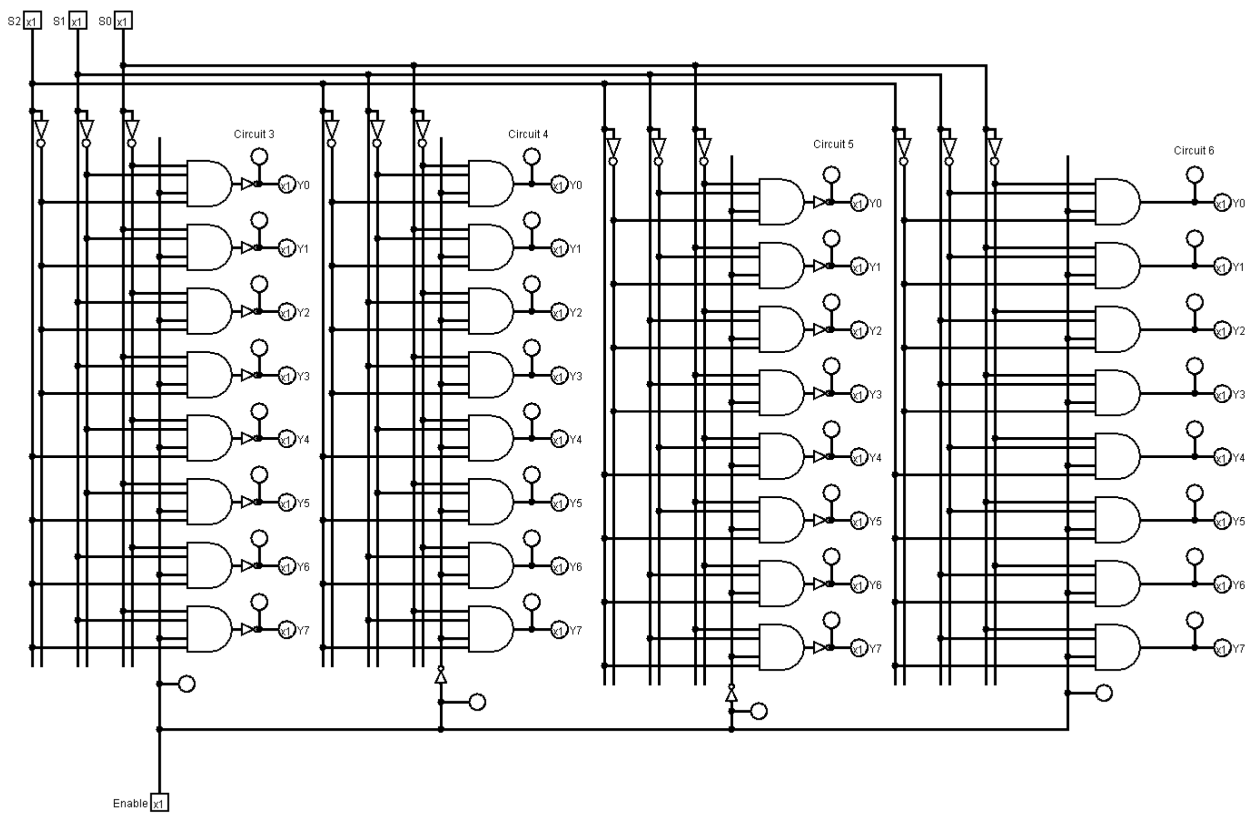


Figure 1. Exploring multiple decoders

Working on their laptops with their neighbors, students download a *Logisim* file containing a set of related circuits and an "exploration guide" (usually as a Microsoft Word file). The "exploration guide" contains a set of instructions (for example "set the inputs to specific values and observe the output", "now change the inputs in a specific way and observe the results", "record your observations here") and a set of questions that focus their attention on different aspects of the circuit. The idea is that the students are "explorers" in a foreign world (the world of digital circuits). The recording of observations (using words, equations and tables) helps students to articulate their findings. One frequent instruction is to write a sentence that summarizes the function of some part of the circuit. It's surprisingly difficult for students to translate key concepts into words, but if they can do so, their thinking seems to be clarified.

The circuit being explored might contain several different specimens (circuits) that behave similarly, and they are asked to identify the differences and classify the specimens. For example, Figure 1 shows one part of the decoder exploration containing all combinations of active high and active low enables and outputs; the actual circuit contains active high and active low LEDs so that the same LED is illuminated in all four circuits at once. Circuits 1 & 2 in the same activity have no enable inputs and introduce the distinction between active high (minterm) and active low (Maxterm) outputs. It is fun to watch them puzzle over the differences and see their pleasure on discovery.

At the end of the decoder exploration is another mystery circuit (a multiplexer) that takes the decoding idea one step further (See Figure 2.) Students are asked to consider two input types (select inputs and information inputs) and to determine the relationship between the input types and the output. With a few hints and loaded questions they can infer the function of the MUX.

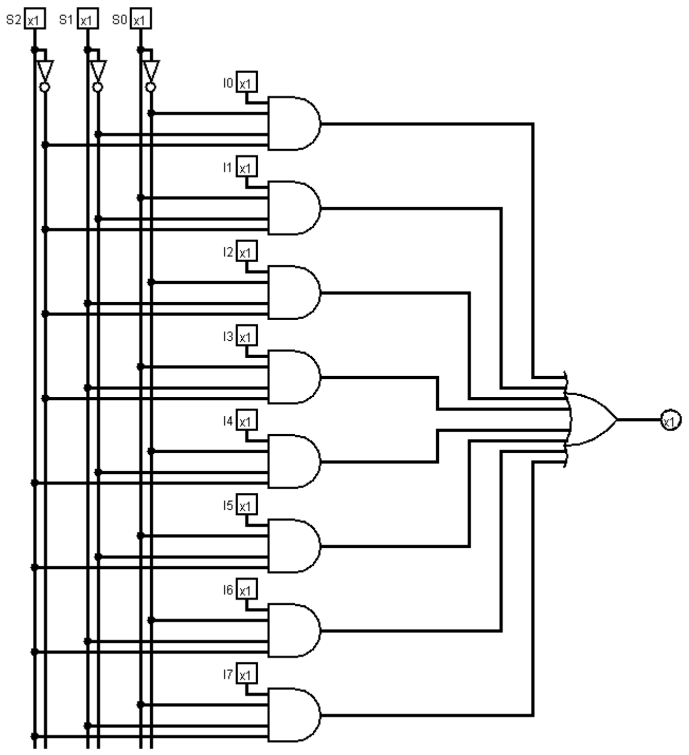


Figure 2. Multiplexer mystery circuit

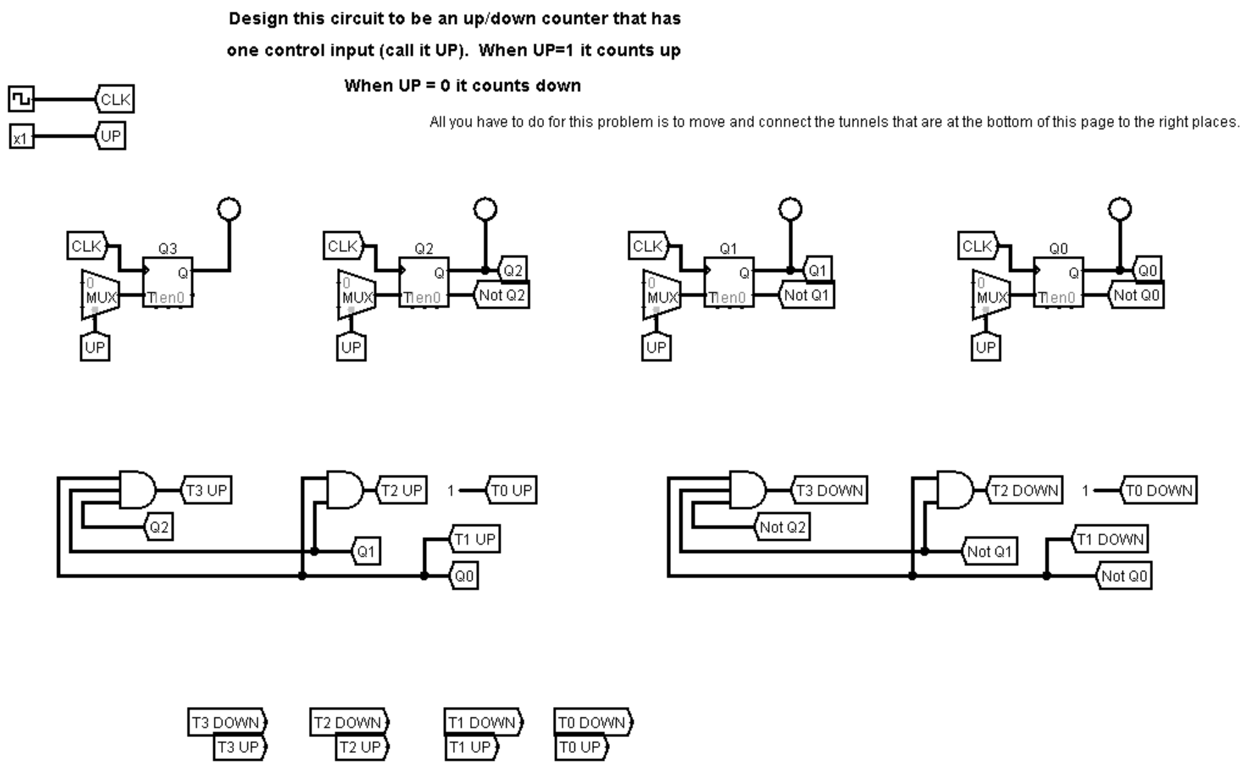
Sometimes they are asked to speculate about what happens if something is changed, and then make the change and observe the results. Other explorations provide a sequence of specimens of increasing complexity so that they can observe the historical "evolution" of circuits. While the students are working on these circuits, the instructor circulates among them; asking questions, providing hints, and encouraging participation. Students frequently ask "what would happen if we did X instead of Y" to which the instructor can answer "try it and see". Thus students are encouraged to try to evaluate their own ideas instead of simply relying on an "expert". Occasionally the instructor returns to the podium to answer common questions or to discuss difficult ideas. The material provided in these short lectures seems to be absorbed and understood more readily since it's providing an answer to a question that they've been pondering on their own.

Towards the end of an exploration class period, the instructor summarizes the findings and encourages continued exploration. In many cases the completed exploration guides are submitted (electronically) before the next class period and a solution (with added explanation) is posted to the class web site. The next class begins by revisiting the exploration and solution, to help ensure understanding. Some solutions include additional "puzzles" or interesting circuits that use the concept that was just explored, so students are "rewarded" (intellectually) for reviewing the solution after they submit their exploration guide. Each exploration also includes some special feature of *Logisim* that they will need for their own designs; revisiting each exploration at the beginning of the next class can draw attention to those special constructs or approaches.

Collecting (and grading) the explorations provides motivation for participation. Weekly quizzes revisit the concepts from the exploration, which tends to reinforce learning. Students are permitted to work with others on the exploration guides, and ample office hours are provided between classes, so the level of engagement remains high.

Karaoke Design

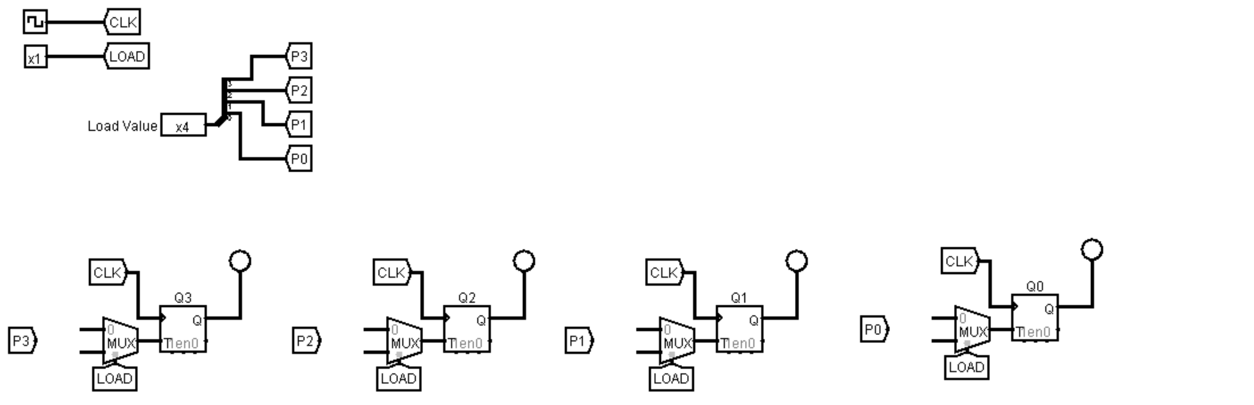
As the semester progresses and the students understand the basic components, we start using another active learning technique, dubbed "karaoke design." The idea of a "karaoke design" is for the students to explore specific design techniques that combine basic components in interesting ways to produce more complex components. There are two learning goals: first we want the students to understand these more complex components (that is, the "product"), as they will use these "products" as components in later designs. Second, we want them to observe and absorb the process by which these components are built. The "karaoke design" activities allow students to interact with both the product and the process.


Design this circuit to be an up/down counter that has one control input (call it UP). When UP=1 it counts up When UP = 0 it counts down

All you have to do for this problem is to move and connect the tunnels that are at the bottom of this page to the right places.

Figure 3. The up/down counter karaoke design activity

In karaoke design, students are provided with a *Logisim* file that contains key components arranged in a suggestive way on the page. They are tasked with connecting the components (and perhaps adding a few simple items) to produce a system to meet a set of given specifications. (In some sense, adding the lyrics to an underlying melody.) The *Logisim* file might contain several circuits that are "stepping stones" towards the desired goal, leading them towards a (sometimes the) correct solution. They "puzzle" over the components that are provided, and are forced to review these components in order to use them in the resulting design. In this way, earlier concepts are continually reinforced and revisited as the semester proceeds.



Determine the inputs to the MUX such that the load value P3 P2 P1 P0 is loaded into the counter when LOAD =0 (active low) and the current value is HELD when LOAD=1

Knowing how to LOAD a T FF is needed for the next part of this assignment.

Figure 4. Karaoke design to load a T flip flop

Extend your up/down counter to add two more functions: CLEAR and HOLD

When PE is ZERO (it is active low) then LOAD an external value P3 P2 P1 P0

When PE is ONE (unasserted) then it COUNTS when CE is ONE and UP determines the direction of counting

IF PE is ONE (unasserted) and CE is ZERO (unasserted) then it will CLEAR if UP is ZERO or HOLD if UP is ONE

Each of these operations has to be synchronized with the clock.
Use the skeleton shown below to implement your multifunction counter.

If PE=1 then
If CE,UP = 0,0 then CLEAR
If CE,UP = 0,1 then HOLD
If CE,UP = 1,0 then COUNT DOWN
If CE,UP = 1,1 then COUNT UP
else LOAD

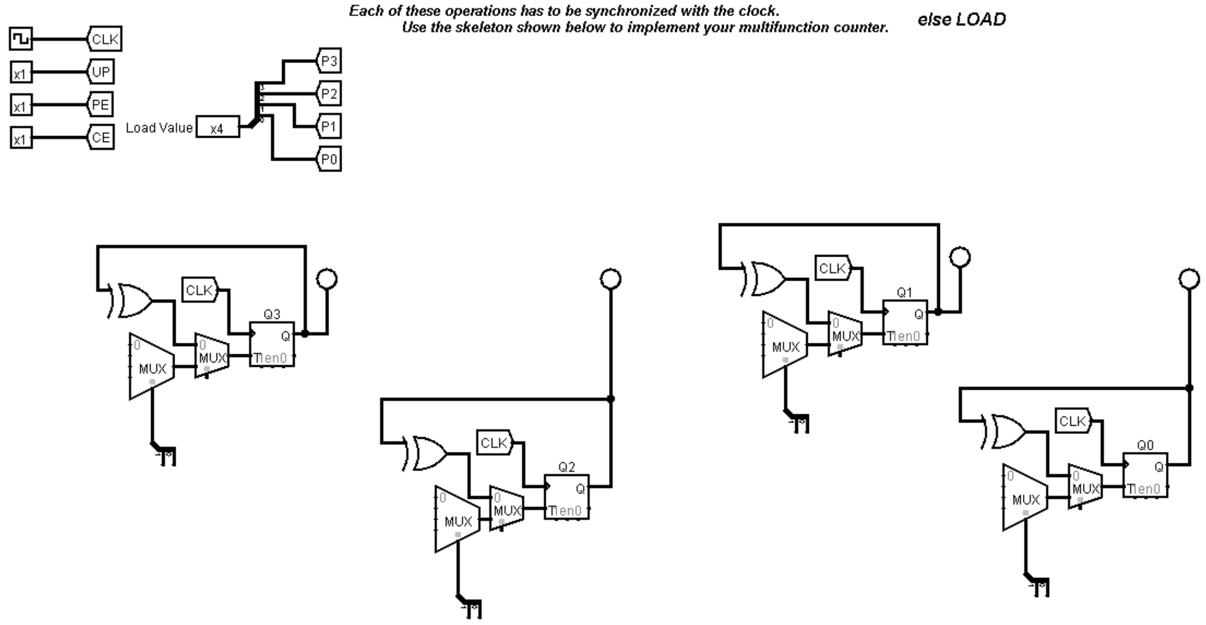


Figure 5. Karaoke design of a multifunction counter

One example karaoke design problem is a multifunction counter (count up, count down, hold, clear and parallel load) using multiplexed inputs to T flip-flops. Students are asked to work in pairs to design (on paper) a 3-bit up counter or a 3-bit down counter using T flip flops (which takes about 10 minutes). They are then given the *Logisim* file that implements (on separate circuits) a 4-bit up counter and a 4-bit down counter. Figure 3 shows a karaoke circuit to combine them to create an up/down counter. Next they are asked to show how to load a specific value into a T flip-flop (Figure 4), which is where they really understand the difference between T and D flip-flops, and on the next circuit they determine how to clear a T flip flop. The final karaoke circuit, which is the goal of this activity, combines all the pieces together and is shown in Figure 5.

Students enjoy the challenge and achieve a real sense of accomplishment when their design works. And they can evaluate whether it meets the specs (i.e. they can test their design) before submitting it. Grading their submissions is time consuming, as there may be multiple correct solutions, but it is easier to test a *Logisim* circuit than it is to evaluate a design that is drawn on paper.

The karaoke design problems are more challenging than the guided explorations and some students grumble about the tedium of making numerous connections. They sometimes claim that they understand the concept and the detail implementation work is "busy work," but in fact the "devil is in the details." These same students are often surprised when their ideas work in concept but not in reality when they need to confront the details. If they are willing to work on the debugging and revision process, they can gain valuable experience. In fact, one of the goals of the course is to help students acquire debugging skills, that is, the ability to determine whether something is working correctly and how to determine the cause of an error when one is recognized. Debugging can be frustrating and time consuming but almost always enhances ones experience base.

As with the explorations, students are permitted to work with others and ample office hours are provided. Explorations are similarly due before the next class meeting at which time the solution is reviewed (the solutions are made available just after the due date). Concepts are often revisited on the weekly quizzes.

LogisimFSM

RTL design combines a datapath and controller in one circuit, using previously-designed components (decoders, multiplexers, registers, counters, etc) into a larger system. At this point in the class, *Logisim* fails us. Many students find it too hard to envision the high-level function of the controller when faced with either a black-box (too many details are visible) or the full implementation (too many details are visible). As an example, consider the soda machine dispenser example from the textbook. (Vahid, 2011) The top-level *Logisim* circuit is shown in Figure 6; the datapath is shown in Figure 7 and the controller is shown in Figure 8. Although the design and implementation of this system is described clearly in the text, it is hard for students to understand the relationship between the states in the controller and the components in the datapath, even as they can simulate the circuit and see the signals pass between the datapath and the controller. This problem is more acute as we move into the design of a programmable processor.

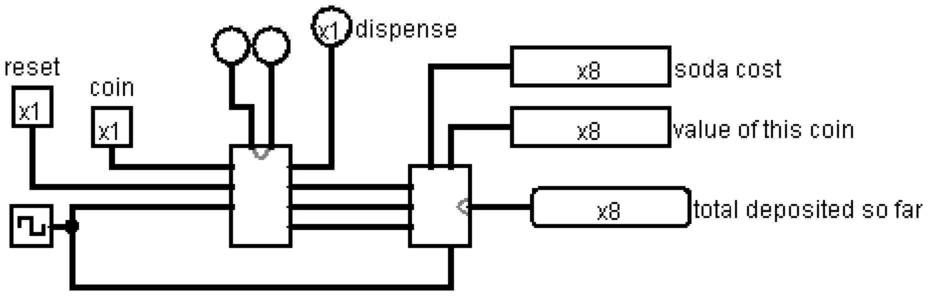


Figure 6. Top-level representation of soda-machine system

Our *LogisimFSM* extension to *Logisim* adds one component type to the pallet: a state for a finite state machine. A state is a square with 4 types of connections: input arcs at the bottom, output arcs at the top, select inputs at the left and an indicator variable at the right. The select inputs define the selection of the next state and the indicator variable for a state is used to produce output signals that drive the datapath. Figure 9 shows the FSM for the soda machine example; this FSM matches the example in the textbook. The datapath and controller are combined on one circuit and students can see the system advance from state to state and control the datapath.

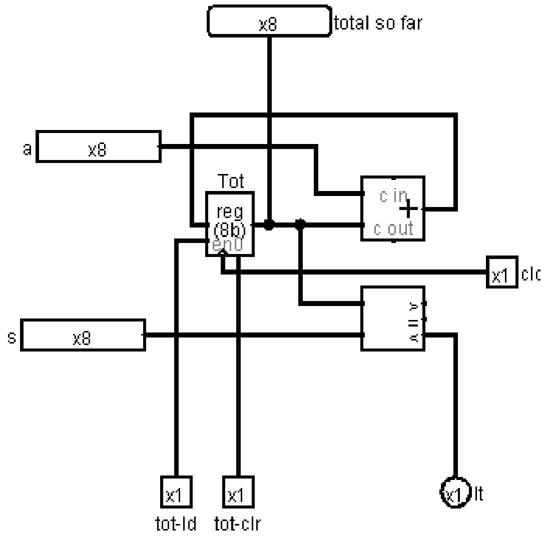


Figure 7. Datapath for soda-machine system

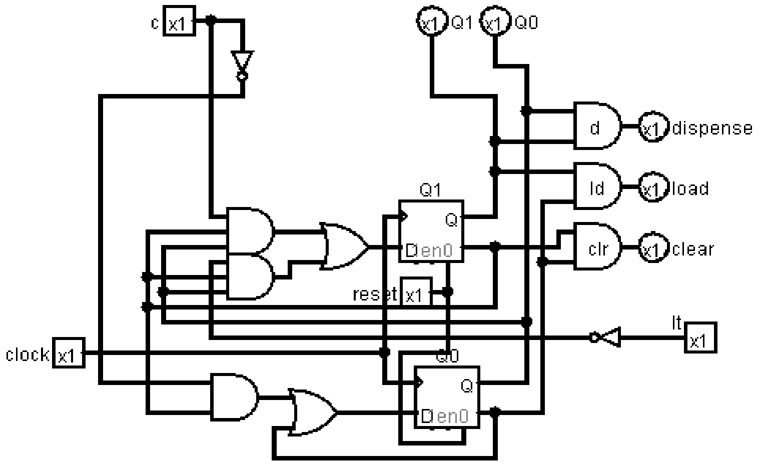


Figure 8. Implementation of controller for soda-machine

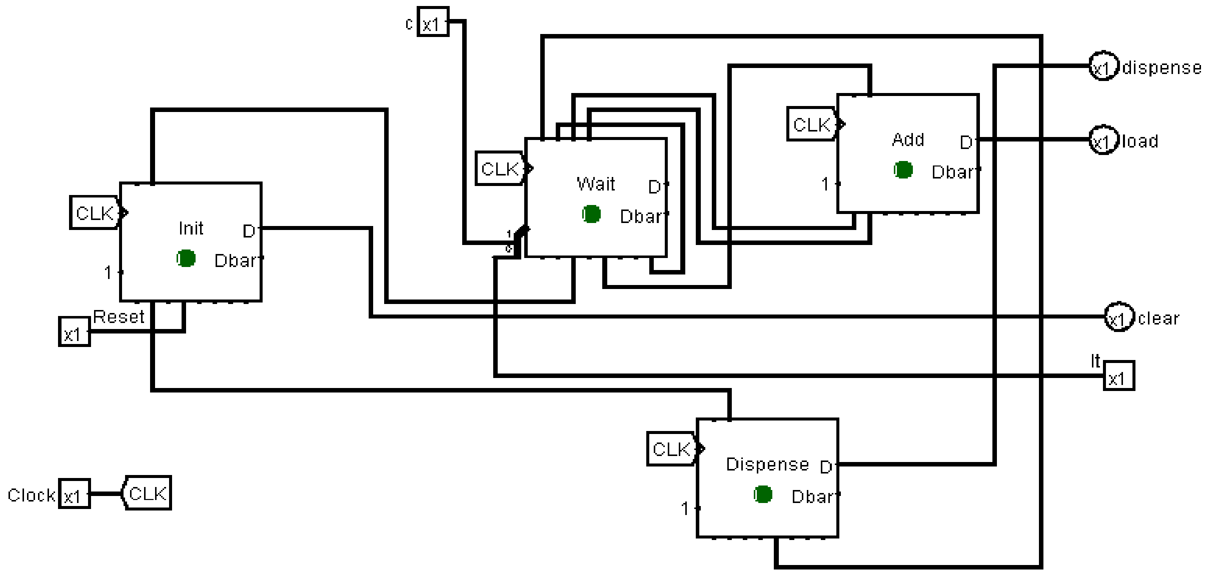


Figure 9. LogisimFSM controller for soda-machine

Once students have been introduced to the state machine construct in *LogisimFSM* they can explore and design systems at the register transfer level (RTL). A *LogisimFSM* implementation of a hardware queue is explored in one class and students transform it to a stack as an assignment. Although some students have seen queues (FISO) and stacks (LIFO) in a computing class, they are pleasantly surprised to see a hardware implementation.

The LogisimFSM program is most effective on the last topic of the class: the programmable processor (Figure 10). The textbook (Vahid, 2011) describes a 3-instruction processor in some detail but the “flat” (on paper) circuit designs are hard to grasp. The implementation of the same 3-instruction processor in *LogisimFSM* allows students to see the fetch-decode-execute cycle in a simulation and to examine the signals that pass between the datapath and controller.

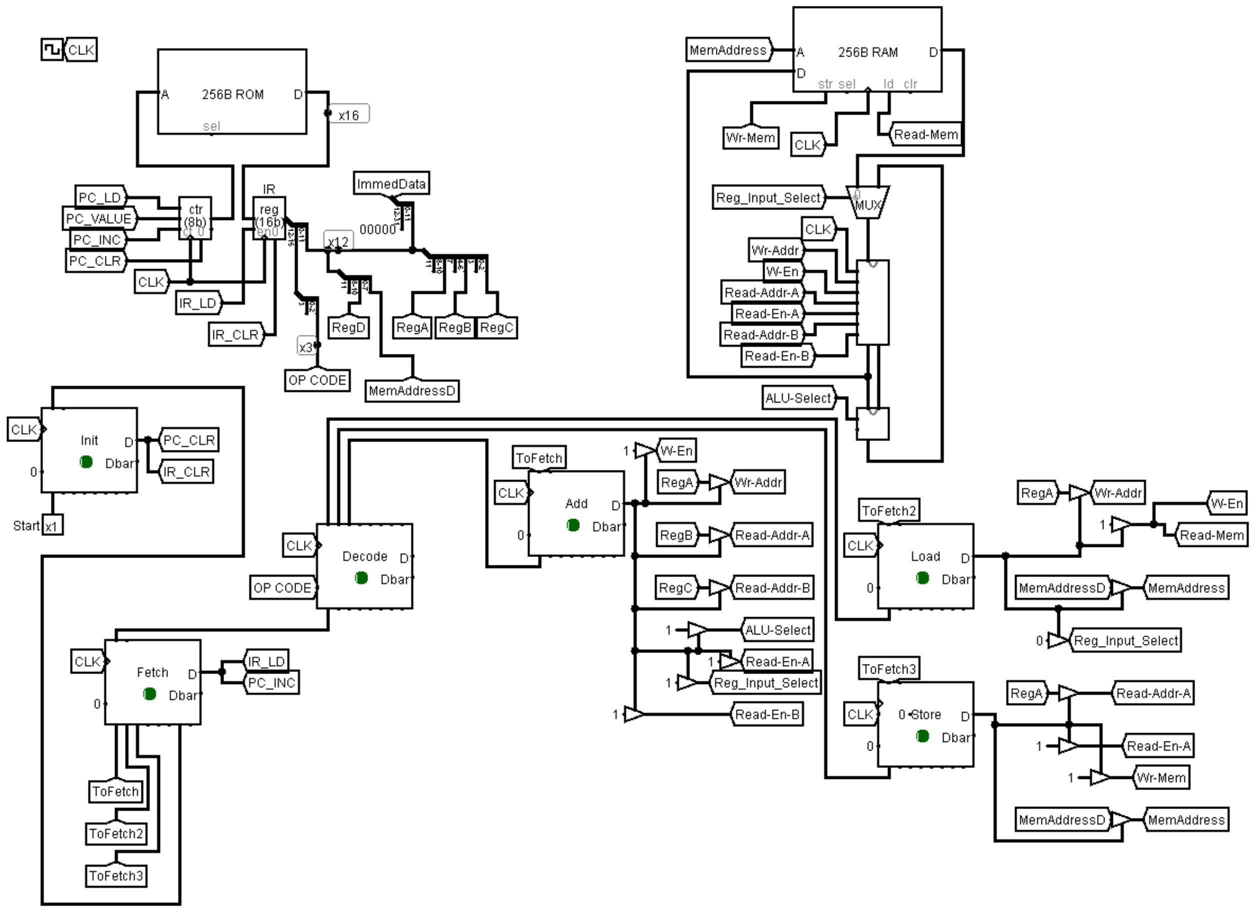


Figure 10. LogisimFSM model of 3-instruction processor

The last karaoke-design assignment in the class is to expand the 3-instruction processor to a 6-instruction processor. The textbook describes the addition of a subtract instruction, a load-immediate instruction and a conditional branch. All the details are given in the text and a skeleton *LogisimFSM* design file includes the additional states to match the exposition in the book. In reality, all the students need to do is look closely at the textbook and translate the material that is described therein into the *LogisimFSM* circuit. The skeleton file contains an actual program (again from the text) in the instruction memory. This enables the students to see and understand how a program is executed at the detailed hardware level. Many students seek help during office hours on this assignment, but the intellectual satisfaction of the students as they “get it” is reward for their efforts.

Pacing changes

About 80% of class meetings included either a guided exploration or a karaoke design. A few classes were lecture based, for example when a completely new topic was introduced or when a specific design process was taught. Even for these few classes, active learning techniques were included as much as possible. In many cases handouts (on paper) were distributed at the beginning of class. These handouts were designed to facilitate note-taking and included templates to accompany each step of the design (structures for tables, grids for matrices, etc.) as well as completed circuit diagrams. These handouts tended to help students structure and organize their notes (and indeed their thought processes) and helped prevent errors. The instructor used expanded versions of the handouts (spread across multiple pages for example) as a template for writing notes (using a smart podium and writing onto pdf files using electronic ink). After class, the instructor's notes were posted to the class web site.

Guided explorations were used more heavily in the earlier part of the semester while karaoke designs were used more in the latter part. Not all explorations were collected and graded; on average we collected and graded one per week. There were also weekly quizzes, biweekly lab assignments and three tests, in addition to the final exam. The explorations and karaoke designs explicitly counted for 10% of the grade, but they also influenced the weekly quizzes (also 10%). Labs counted 25%, tests 30 % and the final 25%. There were a total of 37 graded items per student over the course of the semester.

Impact

It was difficult to directly measure impact for several reasons. First, the course already had a high level of satisfaction (see Table 1) and a reputation for being fun. Second, attendance is not recorded and engagement is difficult to measure. However there were several notable changes. The atmosphere in the class was more "alert" and even the students in the back tended to participate. More students came to office hours than in previous semesters and they asked questions that exhibited deeper understanding of the material. Students were often seen working in small groups in the study nooks of the building which houses the classroom and the author's office. Students "hung around" more after class to ask about the material or to talk to each other. There were times when I had to "shoo" them out so that the next class could begin (we could continue the discussions in the common area outside the classroom). Students frequently continued working on *Logisim* files in the common area outside the classroom after class ended. In fall 2014, one question was inserted on the course evaluation form asking the degree to which students agreed with this statement "The explorations enhanced my understanding of the material". Sixty percent strongly agreed; thirty-five percent agreed; four percent were neutral and one person strongly disagreed.

Student Performance

Although lectures were nearly completely eliminated in favor of active learning materials, we were able to cover more material during the semester. Further, we tolerated three snow days in spring 2014 with these activities. On at least 2 snow days, explorations were amended in the early morning hours to provide more detail or more intermediate steps and posted to the class web site as assignments that substituted for class meetings. This enabled us to keep up with the material even when classes were cancelled.

The most substantive measure of success came in the last two topics, the ones that had been so hard to convey in semesters past. The use of the *LogisimFSM* materials enabled much deeper understanding of those topics. The final exam doesn't change much from semester to semester (the specific questions change but the format and coverage does not). Student performance on the tests and final exam was statistically identical to past semesters with one notable difference.

More in-depth problems were given on those two troublesome topics. So **the students performed comparably on a more difficult exam.**

Finally, it is worth noting that the instructor won three different teaching awards this year, which can be taken as a tangible demonstration of student satisfaction.

Table 1. Summary of student evaluations

Semester	enrollment	Overall Course Rating (out of 5)	Overall Instructor Rating (out of 5)	Comments
Fall 2011	89	4.31	4.6	Traditional lecture with some in-class activities using paper; some use of *Logisim*
Fall 2012	117	4.11	4.52	
Spring 2013	141	4.11	4.48	Increased use of *Logisim* in class; *LogisimFSM* developed (but not used in class)
Fall 2103	125	4.12	4.47	Explorations and Karaoke Design activities developed and refined and used extensively in class; RTL design activities use *LogisimFSM*
Spring 2014	153	4.21	4.55	
Fall 2014	140	4.26	4.71	

Reflections

The materials described in this paper will be used and refined in the coming semesters. These materials have been organized and annotated on a collaborative web site to facilitate use by other instructors (both here and elsewhere). In addition to achieving better understanding of the most difficult topics, we had fun. Students reported great satisfaction both to me personally, on the course evaluations, and on award nominations. Several students decided to change their major (to computer engineering) on the basis of this course. And although I wish that enrollments were more manageable, we've tolerated the increases while still improving the course.

The development and use of the guided explorations, karaoke designs and the new *LogisimFSM* package required a significant time investment. Preparing for each class meeting took as much time as was previously needed for a traditional lecture, even when materials were re-used in the second iteration. Each activity was reviewed and revised slightly to address points that may have been confusing in the past. The ability to refine and improve these materials is a huge benefit of using a software host package (like *Logisim*) rather than static materials (like a book). We did use a book, and many of the explorations and karaoke designs were implementations of examples (or variations) from the text. Students liked to be able to read the book and then interact with the example (via *Logisim*).

One problem remains. A significant number of students seemed unable or unwilling to reflect on the assignments. Once they completed it, they turned it in and moved on to something else.

Occasionally a student would come to office hours with a question whose answer was in one of the explorations. It would be good to develop a way to encourage/force students to revisit the materials and activities, and reflect on the purpose of each one.

There is a large group of materials that were developed for this project. A collaboration site was created with folders arranged by topic. Class notes, handouts, *Logisim* files, exploration guides, karaoke design problems and solutions are all cataloged and annotated. Another instructor could use the materials as they are, or could refine/adapt them as needed (the Microsoft word files are provided), however, significant effort would be required to learn *Logisim* and understand the subtleties of each example. The materials (circuits, exploration guides, assignments, solutions, notes) are all available to other instructors upon request.

Acknowledgement
These materials were developed with the support of a small grant (2 weeks summer salary) from the university teaching resource center. No other funding was available to enable a more comprehensive evaluation.

References

Alaraje, N., Sergeyev, A., & DeGroat, J. (2011). Collaborative Curriculum Development of an Industry-Driven Digital Logic Design. *Proceedings of the 2011 ASEE Annual Conference.* Vancouver, B.C, Canada.

Bill Carroll, S. G. (2014). A Hierarchical Project-based iIntroduction to Digital Logic Design Course. *Proceedings of the 2014 ASEE Annual Conference.* Indianapolis, IN.

Burch, C. *Logisim.* www.cburch.com/logisim.

Carroll, B., Geiser, S., & Levine, D. (2014). A Hierarchical Project-based Introduction to Digital Logic Design Course. *Proceedings of the 2014 ASEE Annual Conference.* Indianapolis. IN.

Carroll, C. (2012). Teaching Digital Design in a Programmable Logic Device Arena. *Proceedings of the 2012 ASEE Annual Conference.* San Antonio, TX.

Devore, J., & Soldan, D. (2012). VisiBoole: Transforming Digital Logic Education. *Proceedings of the 2012 ASEE Annual Conference.* San Antonio, TX.

Hoffbeck, J. (2014). Using Practical Examples in Teaching Digital Logic Design. *Proceedings of the 2014 ASEE Annual Conference.* Indianapolis, IN.

McCarthy, D., Wright, C., Barrett, S., & Hamann, J. (2010). Student-created laboratory exercises for a Digital Systems Design Course using HDL and PLDs. *Proceedings of the 2010 ASEE Annual Conference.* Louisville, KY.

Peterson, B., & Clark, A. (2012). PRISM: The reincarnation of the Visible Computer. *Proceedings of the 2010 ASEE Annual Conference.* Louisville, KY.

Vahid, F. (2011). *Digital Design.* Wiley.

Wang, G. (2010). Preview, Exercise, Teaching and Learning in Digital Electronics Education. *Proceedings of the 2010 ASEE Annual Conference.* Louisville.

Yilmazer, N., & Sekar, R. (2011). A Project-based Hands-on Digital Logic course. *Proceedings of the 2011 ASEE Annual Conference.* Vancouver, B.C., Canada.