

New Web-Based Interactive Learning Material for Digital Design

Prof. Frank Vahid, University of California - Riverside

Frank Vahid is a Professor of Computer Science and Engineering at the Univ. of California, Riverside. His research interests include embedded systems design, and engineering education. He is a co-founder of zyBooks.com.

Dr. Alex Daniel Edgcomb, Zybooks

Alex Edgcomb finished his PhD in computer science at UC Riverside in 2014. Alex has continued working as a research specialist at UC Riverside with his PhD advisor, studying the efficacy of web-native content for STEM education. Alex also works with Zyante, a startup that develops interactive, web-native textbooks in STEM.

Dr. Susan Lysecky, Zybooks

Susan Lysecky received her PhD in Computer Science from the University of California, Riverside in 2006. She served as a faculty member at the University of Arizona from 2006-2014. She has a background in design automation and optimization for embedded systems, as well as experience in the development of accessible engineering curricula and learning technologies. She is currently a Senior Content Developer at zyBooks, a startup that develops highly-interactive, web-native textbooks for a variety of STEM disciplines.

Prof. Roman Lysecky, University of Arizona

Roman Lysecky is an Associate Professor of Electrical and Computer Engineering at the University of Arizona. He received his B.S., M.S., and Ph.D. in Computer Science from the University of California, Riverside in 1999, 2000, and 2005, respectively. His research interests focus on embedded systems, with emphasis on runtime optimization, non-intrusive system observation methods for in-situ analysis of complex hardware and software behavior, data-adaptable system, and embedded system security. He was awarded the Outstanding Ph.D. Dissertation Award from the European Design and Automation Association (EDAA) in 2006 for New Directions in Embedded Systems. He received a CAREER award from the National Science Foundation in 2009 and four Best Paper Awards from the ACM/IEEE International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS), the ACM/IEEE Design Automation and Test in Europe Conference (DATE), the IEEE International Conference on Engineering of Computer-Based Systems (ECBS), and the International Conference on Mobile Ubiquitous Computing, Systems, Services (UBICOMM). He is an inventor on one US patent. He has coauthored five textbooks on VHDL, Verilog, C, C++, and Java programming. His recent textbooks, published with Zyante, utilize a web-native, interactive, and animated approach that has shown notable increases in student learning and course grades. He has also received multiple awards for Excellence at the Student Interface from the College of Engineering at the University of Arizona.

New Web-Based Interactive Learning Material for Digital Design

Frank Vahid, Computer Science & Engin., Univ. of California, Riverside (also with zyBooks)

Alex Edgcomb, zyBooks, Los Gatos, California (also with UC Riverside)

Susan Lysecky, zyBooks, Los Gatos, California

Roman Lysecky, Electrical and Computer Engineering, Univ. of Arizona (also with zyBooks)

Abstract

Studying college-level digital design involves both the learning of concepts and the developing of skills, as with various other engineering topics. For example, converting a Boolean function to a minimized circuit not only involves learning concepts of Boolean algebra, but also developing skills in creating truth tables, minimizing equations via K-maps, and converting equations to circuits. Sequential design involves not only learning concepts of state machines, but also developing skills in describing behavior as a state machine, and converting to a controller. Developing skills especially benefits from practice, via homework. But, assigning and grading sufficient homework problems is a challenge for instructors, due to limited time and resources. Furthermore, traditional grading of homework has long feedback cycles, which is not conducive to learning.

We thus created new web-based interactive learning material for digital design, to replace existing textbooks. To help master concepts, the material makes extensive use of interactive activities like animations and learning questions. To help master both concepts and skills, the material integrates web-based simulators, and a homework system that auto-generates exercises, and that immediately auto-grades student answers while also providing feedback.

The material has been used at over 40 universities and several thousand students thus far. This paper describes the various items used throughout the material to help students both learn concepts and develop skills.

Introduction

College courses on digital design, also called logic design or digital systems, continue to be required by programs in computer engineering, electrical engineering, and computer science. Studying digital design provides students with a solid understanding of how computers work from the bottom up, and forms the basis for study of computer architecture, both of which enhance student confidence and maturity, and can lead to better software designers who understand hardware resources. And, some students embark on careers involving digital design, perhaps creating interface logic, custom accelerator logic, or new general-purpose computing

architectures, using custom/semi-custom chips or field-programmable gate arrays (FPGAs).

Like many computing and engineering topics, studying digital design involves a combination of learning concepts and developing skills. For example, students must learn the concepts of Boolean algebra that involve various properties, and students must develop skills in applying those properties to minimize logic equations. Active learning, such as answering questions or creating designs, is known to be more effective than less-active learning, such as listening to long lectures or reading numerous pages of text⁴. Active learning is more effective not only for developing skills, but also for learning concepts.

Traditional textbooks essentially represent a passive learning experience; the reading and viewing of numerous pages of text and figures is akin to listening to a lengthy lecture. As such, students (especially today's students, but past students as well) often don't learn well from reading textbooks. In part for that reason (along with other reasons like rising prices), our analyses and those of others show that about a third of students today don't even acquire the "required" textbook in a digital design course, and those that do don't make extensive use of the book.

Furthermore, instructors are finding that grading assigned homework is becoming even harder than before, as grading resources have shrunk (especially since the budget crises starting in 2008). Attempts to deal with such shrinking resources tend to be ineffective; not grading homeworks and instead telling students the homework is for their own good leads most students to not do the homework, and grading only a subset of homework problems results in many disgruntled students upset that their work was "wasted" or their low score is based on just a couple non-representative problems. Also, even with plenty of grading resources, the long time feedback loop reduces the learning opportunity, since students don't realize mistakes until graded items are returned several days (or even weeks) later, when the student has moved on to other material.

As a result of the above problems, we embarked in 2014 on creating new digital design material, utilizing the power of the web to create interactive material. This paper summarizes several features of the interactive material for key digital design topics: Boolean algebra, combinational circuits, K-maps, finite state machines, high-level state machines, and datapaths. The paper describes an integrated homework system built around those features. The interactive material of course covers digital design topics not included in this paper for space reasons, such as creating sequential circuits to implement FSMs.

Having its roots in 2014 rather than in the 1980's when digital design dealt with much smaller circuits, the material also strives to teach a modern view of digital design, going bottom up as

before, but being sure to reach register-transfer-level design as promptly as reasonable.

Boolean algebra

Digital design is based on the mathematics and properties of Boolean algebra. Gaining understanding of and experience with those properties is among the first learning tasks. As such, we developed a new web-based tool to allow a student to immediately apply the properties of Boolean algebra to equations. The tool, shown below, presents a palette of properties, and then auto-generates a goal equation and an initial equation. The student must apply properties one-by-one to convert the initial equation into the goal equation. Another tool version has more properties, and allows the properties to be applied left-to-right (as below) or right-to-left as well. The tool is integrated within the web-based learning material. A homework system auto-generates successively harder problems for a student to solve. Scores per student are reported to the instructor.

Figure 1: Boolean algebra tool. (a) User prompted to select a property. Goal and initial equation shown. (b) User selects terms. (c) Continue applying properties to reach goal.

(a) The interface shows a 'Properties' palette on the left with three options: Distributive ($ab+ac = a(b+c)$), Identity ($a \cdot 1 = a$), and Complement ($a+a' = 1$). On the right, the 'Goal: $w = x$ ' and 'Initial: $w = xy+xy'$ ' are displayed. A prompt 'Select a property.' is shown below the initial equation.

(b) The 'Distributive' property is selected, and the initial equation is updated to $w = xy+xy'$. A prompt 'Select the 1st term.' is shown. The 'Identity' property is then selected, and the initial equation is updated to $w = x(y+y')$. A prompt 'Select the 2nd term.' is shown.

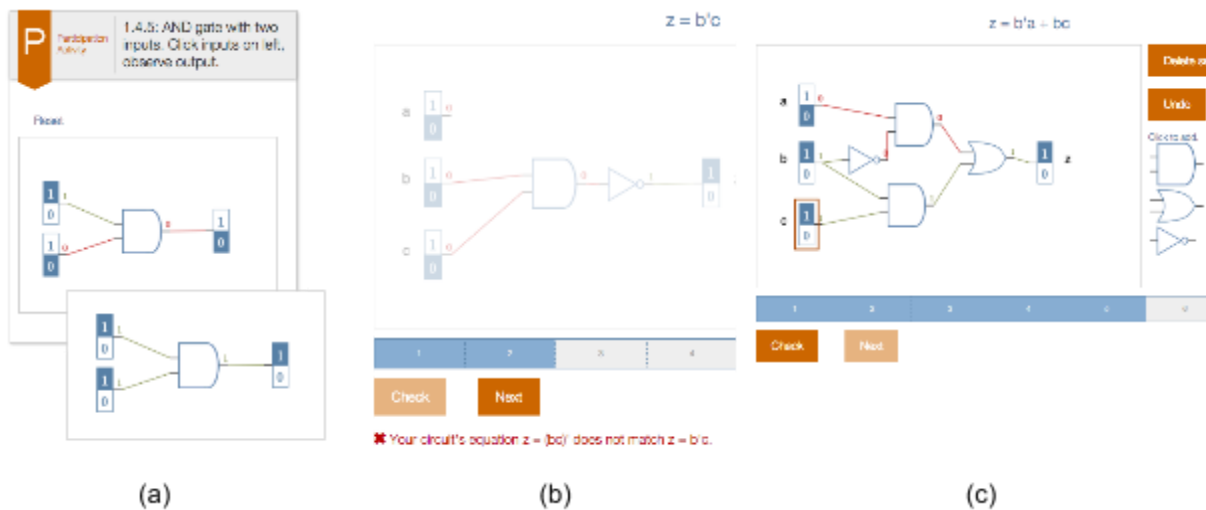
(c) The 'Complement' property is selected, and the initial equation is updated to $w = x \cdot 1$. The final goal is reached, and the word 'Done.' is displayed.

Combinational circuits

A student next learns how logic gates that implement Boolean algebra's operations of AND, OR, and NOT can be connected as combinational circuits to implement Boolean functions. Normally,

a student would learn Boolean functions and combinational circuits conceptually in lecture and via homeworks, while any practice is done separately in labs having circuit simulation tools. Instead, we developed a combinational circuit simulator, shown below, that executes in the web browser. Nearly every example circuit, starting with when an AND gate is first introduced, is shown not as a static figure, but loaded in the circuit simulator. Thus, students can immediately see how the circuits operate dynamically. Additionally, homework problems are built on the simulator: A Boolean function is auto-generated, and the student must create an equivalent circuit, with equivalence being automatically checked. If the student's circuit is wrong, the difference is shown, and then another function of the same complexity is auto-generated. If the student's circuit is correct, a progressively harder function is generated. Such a "progression tool" is a common pattern in the homework problems in the material, typically with about 5 progression levels.

Figure 2: Combinational circuits. (a) Circuit simulator, (b) Auto-generated, auto-graded homework built on top of the simulator, with (c) increasingly harder problems.

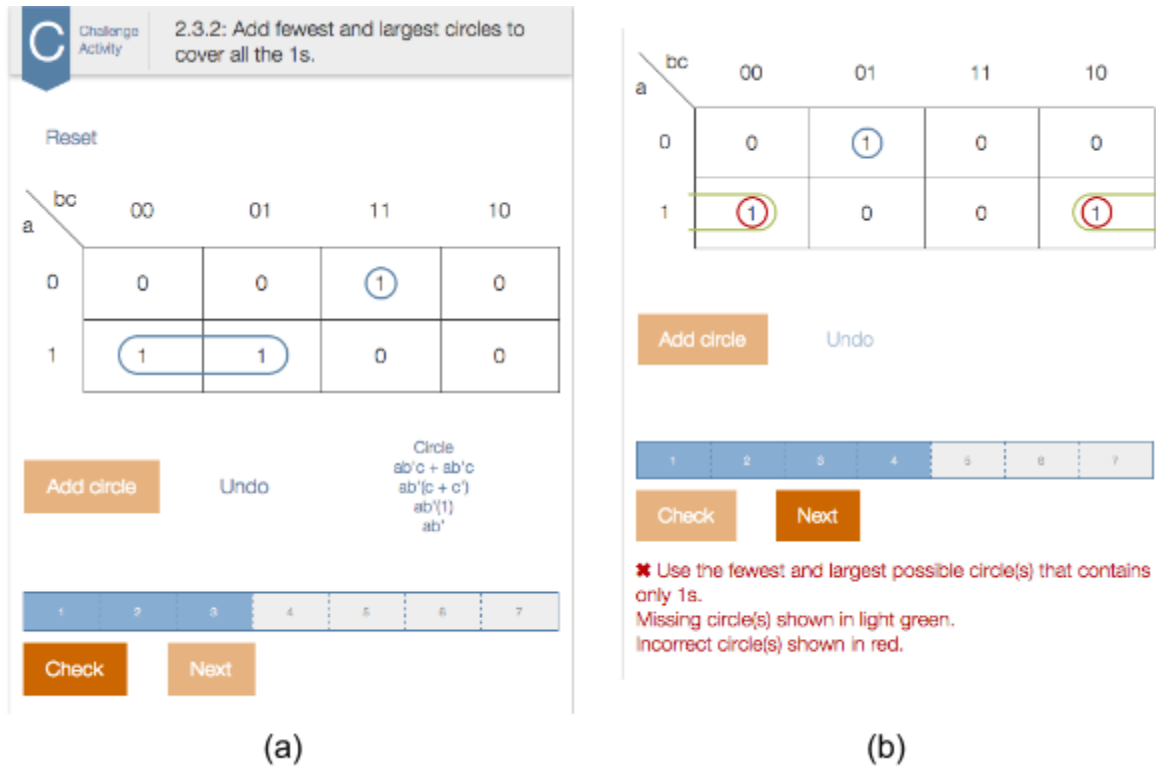


K-maps

K-maps are a graphical technique for minimizing two-level logic. While no longer a job task, K-maps are quite useful for learning, giving students insight on how algebraic techniques are applied to yield minimized circuits. We developed several K-map progression tools for homework purposes, including finding minterms on a K-map, and writing the term that corresponds to a cell or circle on a K-map. Below is the most advanced of those tools, which asks a student to create a minimized two-level equation by drawing the fewest and largest circles. The tool starts with simple problems like a single 1 on the K-map, and presents progressively

harder problems like having 1's that wrap left-right or having three adjacent 1's (so the middle 1 should be included in two circles). The tool automatically shows how a circle corresponds to a series of applied algebraic properties. If the student didn't draw circles yielding minimized equations, the tool shows the correct circles.

Figure 3: Minimization via K-maps. (a) After student fills in the K-map, the corresponding algebraic minimization is shown. (b) The corrected K-map is shown if the equations are not minimized.



Finite state machines (FSMs)

An FSM describes desired behavior of a sequential circuit, just as an equation describes desired behavior of a combinational circuit. Students often have difficulty understanding FSMs because an FSM represents a dynamic thing, but is usually just shown statically. Instructors try to show an FSM's dynamic nature by pointing to (or coloring) one state at a time in lectures. We built a browser-based FSM simulator. Many examples are shown in the simulator rather than as a static figure, so that students can simulate the FSM to see dynamic behavior (and alter the FSM to see the impact). A progression tool is also built on the simulator to provide homework problems, where the student is asked to create an FSM having particular behavior. Problems are auto-generated and get progressively harder, and are auto-graded.

Figure 4: FSM example shown in the simulator with suggested extensions.

Simulate the FSM and note that no output pattern is generated. Then, set a to 1 and note the output pattern is generated.

Extend the FSM so that when an input d is 1, the output pattern moves in reverse. Hint: The transition condition between m and n should be ad', and a new transition is needed from m to p, among other changes.

Figure 5: FSM progression tool (homework).

Generate the repeated pattern 0 1 0 on z

✖ See table.

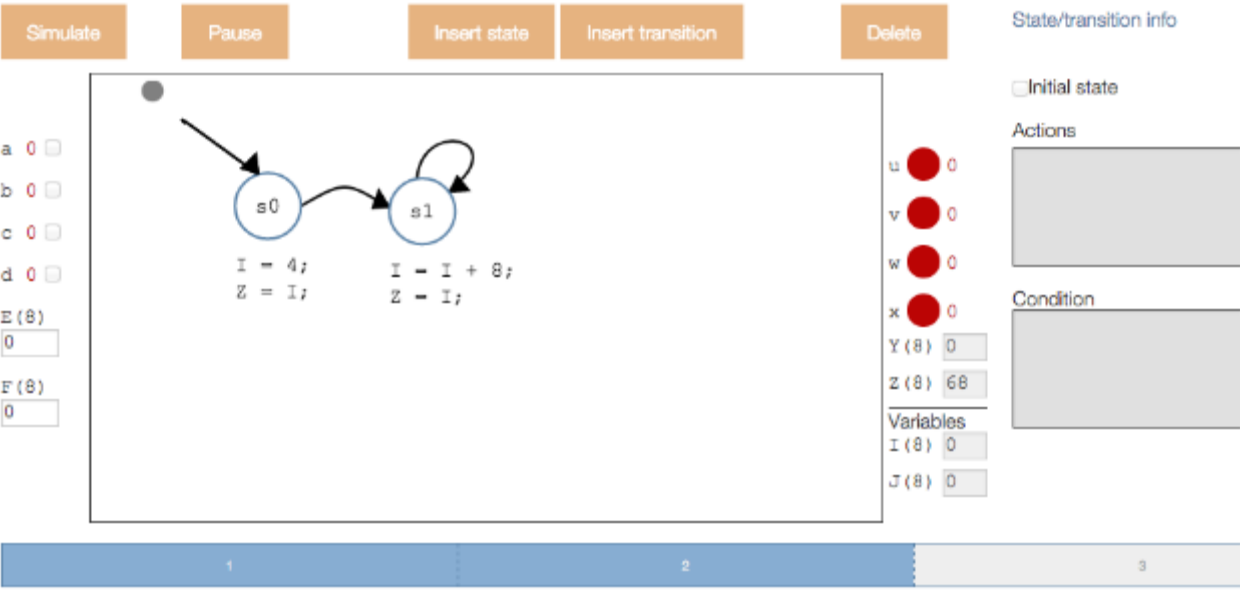
Expected z	0	1	0	0	1	0	0	1	0
Your z	0	0	1	0	0	1	0	0	1

High-level state machines (HLSMs)

Being written for modern digital design, the material includes coverage of register-transfer level (RTL) design. An HLSM describes the designed behavior of an RTL design. The material includes an HLSM simulator, similar to the FSM simulator, as well as a progression tool built on the simulator for homework.

Figure 6: HLSM simulator.

Generate on Z the pattern starting at 4 and incrementing by 8 each cycle.



Simulate Pause Insert state Insert transition Delete

State/transition info

Initial state

Actions

Condition

Variables

u 0

v 0

w 0

x 0

Y (8) 0

Z (8) 68

I (8) 0

J (8) 0

1 2 3

Check Next

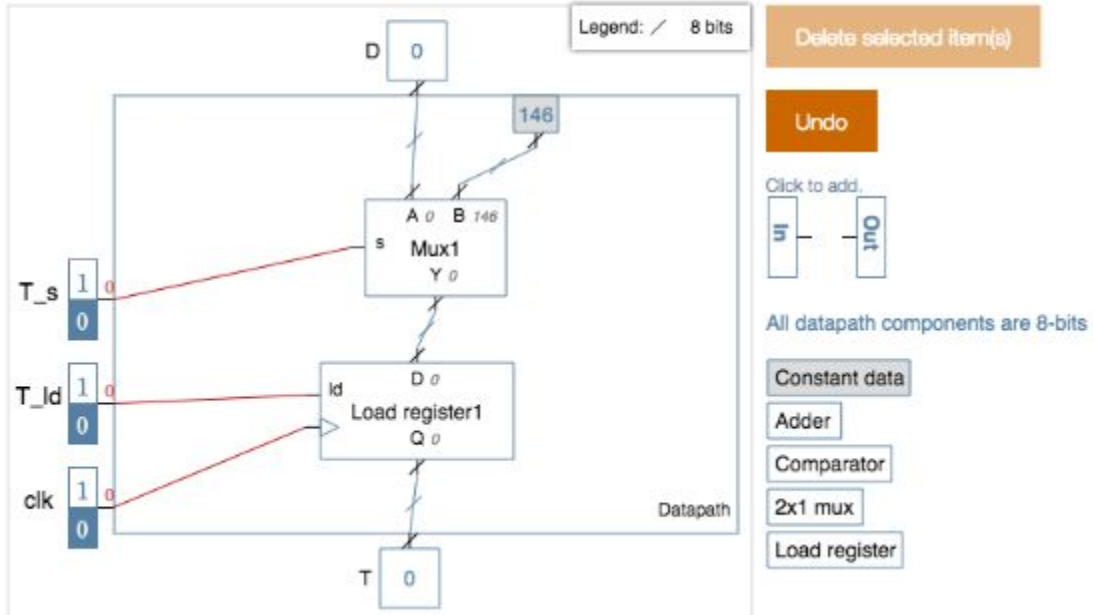
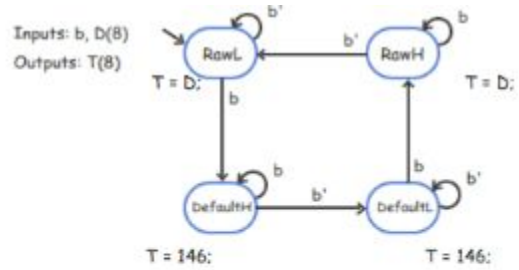
✓ Correct.

Datapath building

A key part of RTL design is designing a datapath able to support an HLSM's data operations (after which the HLSM can be converted to a controlling FSM). Thus, we developed a datapath builder. The resulting datapath can be simulated by clicking on the control and clock input values on the left. A progression tool is built on the simulator, for homework purposes.

Figure 7: Datapath simulator.

Build the datapath for the given HLSM



HDL simulation

Many digital design courses introduce hardware description languages (HDLs), like Verilog or VHDL. In the future, we hope to incorporate a full HDL simulator, likely via a window to a remote server (as we do for learning materials on C, C++, Java, Python, and MATLAB). Presently, we include a constrained simulation for specific topics, like shown below for a combinational process involving a single assignment statement.

Figure 8: Mini HDL simulation.

P Participation Activity

8.2.8: Combinational logic simulator: Designer provides input values; running the simulator generates output values.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity CombinationalLogicExample is
    port(a, b, c : in std_logic;
         z : out std_logic);
end CombinationalLogicExample;

architecture Behavior of CombinationalLogicExample is
begin
    process(a, b, c) begin
        z <= a and (b or c) ;
    end process;
end Behavior;
```

Inputs: click waveform to edit Load sample inputs

a

b

c

0 1 2 3 4 5 6 7 8 9 10

Run

Output:

z

Auto-generated auto-graded homework

In general, our philosophy has been to shift emphasis away from creating end-of-section exercises and solution manuals, and instead spend (much more) effort to create auto-generated, auto-graded homework activities, mostly in the form of progression tools. Reasons include:

- Illicitly-posted solutions to end-of-section exercises for many textbooks are available online today. Plus, a major college-focused web company created solutions to such exercises, selling them to students, yielding a top revenue source for the company.
- Auto-generated problems tend to be harder to copy among students, since each student gets unique problems. Our analyses also show that students will earnestly attempt well-designed learning activities, without trying to copy answers even when those answers are easily available. The progression tool approach presents simpler problems first and progresses to harder problems, and often provides solutions/hints to wrong answers followed by generating a similar-difficulty problem; as such, the tool "adapts" to the student, providing multiple problems until the student gets the problem right, before progressing to harder problems. Cheating is thus mostly unnecessary.

- Students get immediate feedback. An advantage is that students are immediately informed of mistakes, can learn from them, and correct them. A disadvantage is that students may repeatedly guess until completing the activity.
- Automated homework eliminates much or all homework grading for instructors, freeing instructors to perform higher value activities with students, while also enabling instructors to assign the amount of homework the teacher feels is best for students, without worrying about grading.

Animations, learning questions, and less text

The above discussion focused on the built-in web-based simulators and automated homework problems in the material. In addition, the material has been written natively for the web, and thus has a very different appearance than a textbook ported to the web. In particular, whereas a textbook author's only available teaching mechanisms are text and figures, a web-material author also has available interactive mechanisms like animations, learning questions, and tools. Thus, natively writing for the web results in more subject being taught via those other mechanisms, necessarily resulting in less text and fewer static figures. The result is a rather different-looking "book", which at first glance appears light or supplementary, but instead has much of the learning done in a different form (which our studies show yield improved learning outcomes).

Independently, for any text that is written, we have focused on using the fewest words possible. Such minimal-text writing is harder than writing more text. In fact, often the text is first written in a lengthier form, and then reduced without losing meaning. One might consider the effort that goes into a newspaper headline, to understand how hard minimizing text can be. Minimal text has been shown to improve learning by over 100%³; students are more likely to read minimal text carefully and thus learn the concepts therein.

Below is an example discussion, on the topic of timing diagrams. A few lines of text define the concept. The animation dynamically shows the timing diagram growing to the right (which can't be shown in this static paper, obviously). The learning questions in this case replace what would normally be an example presented as text and a figure, instead engaging the student to complete the various parts and showing a brief explanation for each part; such questions are not quizzes or homework, but rather a replacement for lengthier text. As an analogy, rather than lecturing to the student for 10 minutes (via several pages of text and figures), the material lectures for a few minutes (via minimal text), and then has a dialog with the student (via the learning questions and their explanations). We utilize various types of learning questions: true/false, multiple choice, short answer, and matching. In all cases, explanations are included (as well as hints); especially explaining why a wrong answer is wrong, thus breaking down student misconceptions (which education research has found is crucial to enable learning).

Figure 9: The material teaches through less text, in favor of more interactive animations and learning questions.

1.8 Timing diagrams


A **timing diagram** graphically shows a circuit's output values for given input values that change over time. Each signal (input or output) name is listed on the left. Time proceeds to the right. Each signal is drawn as a high line (1) or a low line (0).

P

Participation Activity

1.8.1: Timing diagram for an AND gate.

1 2 3 4 ▶




Timing diagram

a	1	_____
	0	_____
b	1	_____
	0	_____
Y	1	_____
	0	_____

If a = 0 and b = 0, then y = 0.

1 2 3 4 ▶




Timing diagram

a	1	_____
	0	_____
b	1	_____
	0	_____
Y	1	_____
	0	_____

If a = 1 and b = 0, then y = 0.

1 2 3 4 ◀



Timing diagram

a	1	_____
	0	_____
b	1	_____
	0	_____
Y	1	_____
	0	_____

If a = 1 and b = 1, then y = 1.


Figure 10: Learning questions are not quizzes/homeworks, but a more-engaging/effective replacement for reading.

P

Participation Activity

1.8.2: Logic gates: NOT, AND, and OR.

Complete the timing diagram.



j				
k				
y		(a)	(b)	(c)

#	Question	Your answer
1	(a) ✓ j = 1, k = 0 1 OR 0 = 1	0 <div style="background-color: #6b8e23; height: 15px; width: 100%;"></div>
2	(b) ✓ j = 1, k = 1 1 OR 1 = 1	0 <div style="background-color: #6b8e23; height: 15px; width: 100%;"></div>
3	(c) ✓ j = 0, k = 0 0 OR 0 = 0	0 <div style="background-color: #6b8e23; height: 15px; width: 100%;"></div> 1

Time to develop

The interactive material described above is harder to develop and maintain than traditional textbook material (two of the authors have published traditional textbooks). As one obvious indicator, the material is often developed first by writing text and drawing figures (as for a traditional textbook), followed by an additional stage of replacing some text/figures by an animation, and replacing other text by learning questions. Learning questions especially require much attention. Furthermore, focusing on minimal text requires even more time.

Separately, developing the simulators and progression tools required software development time. Some simulators required about 2-3 days of development, while others required several weeks. Furthermore, building a progression tool on top of a simulator typically requires another 2-3 days per tool. In addition to such software development time, the authors must also contribute time to define and refine each simulator and tool, typically totally about 4-8 hours per item.

Beyond initial development, interactive material is harder to maintain. Software must sometimes be upgraded in response to browser updates, or adjusted to work around browser bugs. Software bugs of course must be fixed. Furthermore, being web-based material, the opportunity exists to receive feedback instantly from users, and we take full advantage of that opportunity⁵. Thus, the material is continually being updated to be clearer, and fix any errors (rarely does more than one person ever see an error).

In short, we estimate that developing the web-based digital design material required about 2x more initial effort (well over 1500 hours), and then incurs ongoing maintenance/upgrades of perhaps 6-10 hours per month on average. However, we believe that such web-based publishing of interactive material represents the future of college-level learning material.

Experiences

The material is published as a zyBook (see zyBooks.com). The material has been used at over 40 universities since being published in 2014, with the number growing. Feedback from instructors and students has been very positive.

A common problem for instructors is that instructors know that students learn through homework, but grading such homework is extremely resource intensive. The above-described material has about 1,000 short learning questions (participation activities), and about 100 auto-generated challenge activities (homework-like problems such as using a K-map for minimizing logic, or drawing a simple circuit or state machine), all auto-graded. Grading such activities manually, as done in the past, would have required approximately 20 sec per learning

question and 1 minute per challenge activity, totalling $1000*0.2 + 100*1 = 300$ minutes of grading time per student. Our class at _____ has about 100 students per class, meaning grading time of $300 * 100 = 30,000$ minutes or 500 hours. However, our teaching assistant time is 300 hours for the entire class, and of course TAs can spend all their time grading either. What thus happened before was that far less homework would be assigned than was desired, totalling about 50 hours for the class' duration. Furthermore, feedback would not be returned until at least a week later, diminishing the usefulness. With the new interactive material, students can be assigned sufficient tasks for before class (participation activities) and after class (challenge activities), and yet no teaching resources are spent on creating or grading such items. Furthermore, our previous analyses show that about 95% of students earnestly work on those activities, as opposed to cheating the system to earn points. Completion rates of the activities by students are nearly 100%.

Several instructors report substantially improved student outcomes. In previous work, we showed via randomized controlled studies that students learn more from such interactive material¹ (16% improvement, with the initially weakest students showing 64% improvement), and via cross-semester analyses showing that classes who switch to such interactive material see student grades improve by about 1/3rd a letter grades². Via surveys, we have found that 1/3 of students previously acquired the "required" textbook, whereas nearly 100% acquire such interactive material (due to activity completion contributing to their grade).

Conclusions

We described new web-native learning material for digital design, intended to replace existing textbooks, as well as to provide an integrated automated homework system. The material is intended to help students learn concepts while also developing skills related to digital design. The auto-grading not only provides immediate feedback, but greatly reduces the teaching resources needed to create and grade homework assignments.

Acknowledgements

This work was supported in part by the National Science Foundation (CPS1136146), the National Science Foundation's Small Business Innovative Research (SBIR) program (1315094 and 1430537), and a Google faculty research award. We are sincerely grateful for their support.

References

- [1] Edgcomb, A. and F. Vahid. Effectiveness of Online Textbooks vs. Interactive Web-Native Content, Proceedings of ASEE Annual Conference, 2014.
- [2] Edgcomb, A., F. Vahid, R. Lysecky, A. Knoesen, R. Amirtharajah, and M.L. Dorf. Student Performance Improvement using Interactive Textbooks: A Three-University Cross-Semester Analysis, Proceedings of ASEE Annual Conference, 2015.
- [3] Edgcomb, A., F. Vahid, and R. Lysecky. Students Learn More with Less Text that Covers the Same Core Topics, Frontiers in Education Conference (FIE), IEEE, 2015
- [4] Freeman, S, S.L. Eddy, M. McDonough, M.K. Smith, N. Okoroafor, H. Jordt, and M.P. Wenderoth. Active learning increases student performance in science, engineering, and mathematics. Proceedings of the National Academy of Sciences, 111.23, pgs 8410-8415, 2014.
- [5] Vahid, F., D. de Haas, S. Strawn, A. Edgcomb, S. Lysecky, and R. Lysecky. A Continual Improvement Paradigm for Modern Online Textbooks, Proceedings of International Conference on Education, Research, and Innovation (ICERI), 2015.

Appendix

The Digital Design zyBook's table of contents is provided below. New chapters and tools are being developed.

- 1. Combinational Logic
 - 1.1 Voltage and current
 - 1.2 Switches
 - 1.3 CMOS transistors
 - 1.4 Transistors and gates
 - 1.5 Boolean algebra
 - 1.6 Boolean algebra and digital design
 - 1.7 Digital circuit simulator
 - 1.8 Timing diagrams
 - 1.9 Equations to/from circuits
 - 1.10 Basic properties of Boolean algebra
 - 1.11 Sum-of-products form for circuits
 - 1.12 Binary and counting
 - 1.13 Sum-of-minterms form
 - 1.14 Truth tables
 - 1.15 Why study digital design
 - 1.16 Basic circuit drawing conventions

- 2. Combinational Logic II
 - 2.1 Two-level combinational logic minimization
 - 2.2 K-maps: Introduction

- 2.3 More K-maps
- 2.4 More Boolean algebra: DeMorgan's Law
- 2.5 XOR and XNOR gates
- 2.6 Universal gates
- 2.7 Muxes
- 2.8 Decoders

3. Sequential Logic

- 3.1 SR latches
- 3.2 Flip-flops
- 3.3 Basic registers
- 3.4 FSMs
- 3.5 FSM simulator
- 3.6 Capturing behavior with FSMs
- 3.7 FSMs to circuits
- 3.8 Circuits to FSMs
- 3.9 Reducing states
- 3.10 State encodings
- 3.11 Mealy FSMs
- 3.12 FSM issues
- 3.13 Controller clock frequency

4. Datapath Components

- 4.1 Adders
- 4.2 Signed numbers in binary
- 4.3 Subtractors
- 4.4 Comparators
- 4.5 N-bit muxes
- 4.6 Load registers

5. RTL Design

- 5.1 HLSMs: Introduction
- 5.2 HLSMs with variables
- 5.3 HLSMs with a loop
- 5.4 HLSM simulator
- 5.5 Capturing behavior with HLSMs
- 5.6 Datapaths for HLSMs
- 5.7 HLSMs to circuits: RTL design
- 5.8 RTL timing
- 5.9 Assigning and reading variables

6. Datapath Components II

6.1 Tradeoffs

6.2 Carry-lookahead adders

6.3 Multipliers (array-style)

6.4 Register files

6.5 Multi-function registers

6.6 ALUs

7. Verilog HDL

7.1 Introduction to HDLs

7.2 Combinational logic

7.3 Identifiers

7.4 Testbench

7.5 Sequential logic

7.6 RTL design

8. VHDL

8.1 Introduction to HDLs

8.2 Combinational logic

8.3 Identifiers

8.4 Testbench

8.5 Sequential logic

8.6 RTL design