Paper ID #27068

Student Usage of Digital Design Interactive Learning Tools in an Online Textbook

Dr. Yamuna Rajasekhar, zyBooks

Yamuna Rajasekhar received her Ph.D. in Electrical Engineering from the UNC Charlotte. She served as a faculty member at Miami University where her research was focused on assistive technology, embedded systems, and engineering education. She is currently a Content Developer at zyBooks, a startup that develops highly-interactive, web-native textbooks for a variety of STEM disciplines.

Dr. Alex Daniel Edgcomb, Zybooks

Alex Edgcomb is Sr. Software Engineer at zyBooks.com, a startup spun-off from UC Riverside that develops interactive, web-native learning materials for STEM courses. Alex is also a research specialist at UC Riverside, studying the efficacy of web-native content and digital education.

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.

Student Usage of Digital Design Interactive Learning Tools in an Online Textbook

Abstract
Digital design is a foundational course in computer science, computer engineering, and electrical engineering that describes the basic logical processes from which digital systems are designed. While digital design courses have typically relied on physical textbooks for learning, many classes have begun using a zyBook, which is a web-native, online textbook that includes interactive learning tools. Learning tools are topic-specific, open-ended activities embedded within a topic section, such as a K-map simplification tool or a finite-state-machine simulator. Students have unlimited time to interact with the learning tool. Such interactions are automatically recorded and instructors can see completion rates of each activity by each student. Some learning tools auto-grade, whereas others are followed by formative assessment questions (e.g., short answer and multiple choice). This paper describes auto-graded learning tools, called challenge activities, from various topics in the Digital Design zyBook. Instructors often assign challenge activities as homework. Each challenge activity has around 2-7 levels that become incrementally harder at each level, where the first level is deliberately simpler and the last level is harder, typically the difficulty of a written homework problem. Each level randomly generates questions from a large set of equally-difficult questions. When a student correctly answers the question, the student can proceed to the next level. This paper describes student usage of 4 challenge activities, across 3,099 to 4,518 students (varies by activity) at 43 universities, to provide insight into how such activities impact the student learning experience in digital design courses. Key insight: Students tend to make small mistakes on early levels, then quickly solve the harder levels, enabling most students to complete a challenge activity within 10 minutes.

1. Introduction

Digital design is a crucial course, often required for Electrical Engineering (EE), Computer Engineering (CE), and Computer Science (CS) majors. EE and CE require the course during lower-division, and CS tend to require the course during upper-division, which causes difficulty for instruction because students have varied backgrounds. Another difficulty with teaching (and learning) digital design is the blend of hardware and software through theoretical concepts. For example, a finite state machine (FSM) may be implemented as a circuit or program code.

Digital design contains many difficult concepts that often require much practice and feedback. The topics tend to be highly visual and ripe for interactivity. For decades, educators and researchers have developed interactive tools and simulators to help students learn digital design. However, few such tools enable a student to receive rapid feedback, such as via auto-grading.

This paper describes various interactive tools, called challenge activities, that support immediate feedback via auto-grading, multiple levels of incrementally higher difficulty, and random question generation. Further, for each interactive tool, we analyzed student submissions, both analyzing key metrics and manually reviewing numerous students' submissions.

2. Background

This section describes others' work on interactive tools for learning digital design topics. This section focuses on the topics covered in this paper's challenge activities to give better context of prior work. Many interactive simulators have been developed for educational purposes; most of which are designed for students to tinker with a concept, rather than as homework. For example, numerous K-map simulators were designed for tinkering [1][2][3][4][5]. One such K-map simulator shows the K-map and associated Boolean expression and truth table [1]. A user can toggle each cell of the K-map via clicking that cell; similarly, a user can click the output of the truth table. Circles around 1s are not drawn on the K-map. A user can choose between a 2-, 3-, or 4-variable K-map, and whether to include "don't care" values. Another simulator shows a K-map and associated Boolean expression [3]. A user can toggle each cell of the K-map via a click, and the circles are immediately drawn. A user can choose between 1-8 variables, whether to include "don't care" values, and whether to show the Boolean expression.

One SR latch simulator shows the SR latch circuit and the associated truth table for inputs and outputs [6]. Clicking a circuit input will toggle that input and update the output variables in the circuit and truth table. On the same webpage is another SR latch simulator that shows just the timing diagram of inputs and outputs. The user can toggle the inputs at each time step, and then click Simulate to see the output. Another simulator leverages a circuit simulator, wherein the SR latch has been pre-built, the user can toggle the inputs, and the user can reconfigure the circuit, though that changes the circuit from an SR latch [7]. Over the last 30 years, numerous open-ended FSM simulators have been developed by the research and education community [8][9][10][11][12][13]. For example, JFLAP was released in 2018 and implemented in Java. JFLAP was designed for learning basic concepts in formal languages and automata theory; however, such a tool may also be useful for introducing FSMs in a digital design course. A user can add states and transitions, as well as simulate. A user must set one state as the initial state and may set a state as the final state. Before simulation, the user can enter a set of inputs that would be consumed during execution. During simulation, the current state is highlighted, and consumed inputs are grayed out. Another example, more specific to digital design, also supports the ability for a student to see the timing diagram and associated truth table [13].

3. Challenge activities

This paper presents a few of the challenge activities in the Digital Design zyBook [14] in detail. The key advantages of these challenge activities are that they are auto-generated and auto-graded [15]. This makes it harder for students to copy as each student get a unique problem. Another key advantage is immediate feedback, this allows the student to learn from a mistake and attempt the activity again, improving overall learning. Challenge activities are a linear sequence of incrementally more challenging levels. Each increment is generally 1 new concept. Students can attempt as much as needed, there is no limit to the number of attempts for a particular activity. The activities are randomized, and when a wrong answer is submitted, the student is given a new randomly-generated problem of equal difficulty. Explanations are included on each level of the activity to show the correct answer, often include a hint, and sometimes explain how to arrive at the correct answer. The challenge activities are HTML5, meaning the activities work in any modern browser and across any device that supports such a browser.

4. Methods

Students in the Fall 2018 term who used the Digital Design zyBook were included. The number of students who worked on a particular Challenge Activity varied between about 1,500 and 4,700. Some courses covered all topics; some covered a subset. We analyzed each challenge activity both by manually reviewing student submissions and computing key metrics. The findings of each are presented in the section of the respective challenge activity. Metrics were computed for each level of each challenge activity:

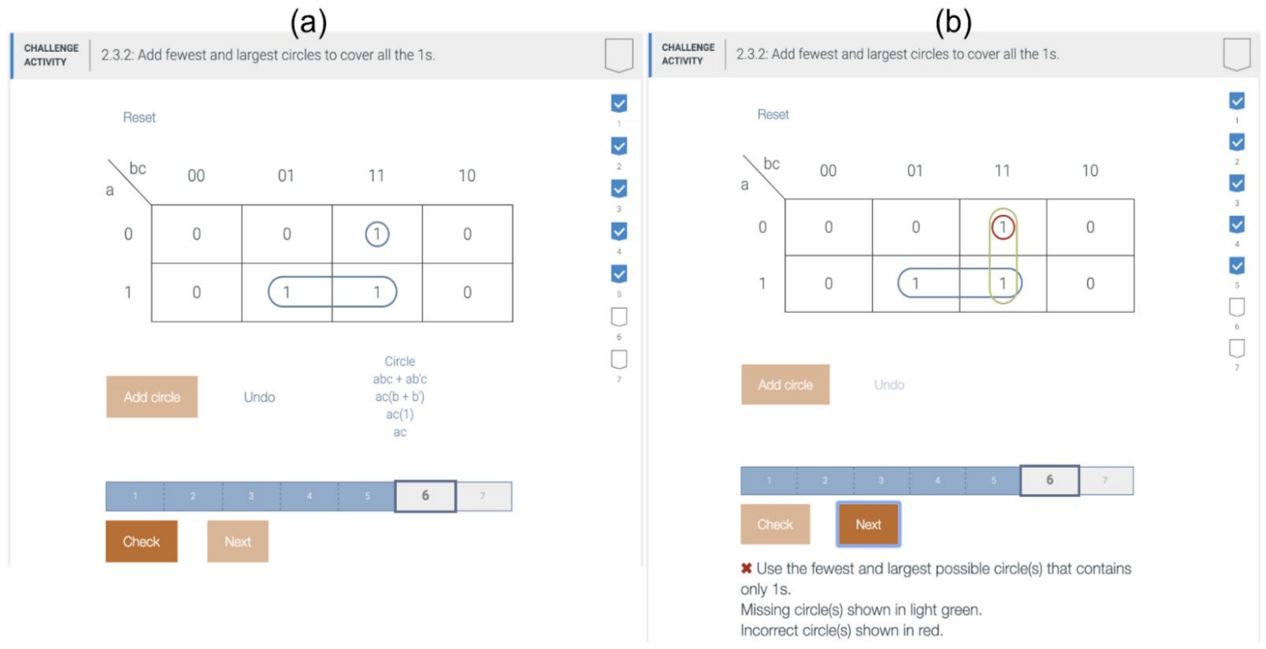
- Time spent: The amount of time before the first correct submission. The time between submissions was summed, if the time between was more than 10 minutes then that time was not summed.
- Number of attempts: The number of attempts until the first correct submission. So, if submitted correctly on the first try, then number of attempts is 1. Note: If a student gave up without making a correct submission, then those attempts are still included here.
- First-time wrong %: The percentage of students whose first submission was incorrect.
- Gave up %: The percentage of students who submitted at least once, but never submitted a correct answer.

5. Challenge activity: Simplify a 3-variable K-map

A K-map is an effective and simple method to reduce equations. K-maps are often taught in the early weeks of a digital design course. The goal of this activity is to learn how to reduce a 3-variable K-map. Before reaching this activity, a student has seen multiple examples of a 3-variable K-map and learned basics like the minterm representation of each cell on the K-map.

Also, the student has had some introductory practice with the rules of circling the 1s. The activity has 7 levels. The first level has a 1 randomly placed in a cell in a 3-variable K-map; that 1 requires a single circle. The second and third level have two 1s randomly-placed adjacently; those 1s also require a single circle. The fourth and fifth levels each have three 1s, two of which are placed adjacently; they require one small circle and one big circle. The sixth (shown in Figure 1) and seventh levels each have three 1s that are contiguous; they require two big circles that overlap. As shown in Figure 1, the K-map challenge activity prompts the student to add the fewest and largest circles to cover all the 1s in the K-map. A student selects cells one-by-one, then clicks the "Add circle" button. If the selected cells are adjacent and the number of cells is a power of 2 (i.e., 1, 2, 4, or 8 cells), then a circle is added around the selected cells. Otherwise, the student is immediately given an error message stating: "Invalid circle. Valid circles can contain 1, 2, 4, or 8 cells." When a student is ready to submit, as in Figure 1(a), the student clicks Check. If the student's submission is incorrect, as in Figure 1(b), then the student is given an explanation, including which circles were incorrect and what the expected circles were. Otherwise, the student progresses to the next level.

Figure 1: K-map challenge activity wherein a student adds fewest and largest circles around the 1s. (a) Student added a circle around two of the 1s, and then added a circle around the third 1 by itself. Also shown is the simplification process for the circle from the full expression to the simplified expression. Then, the student clicks the Check button. (b) The activity highlights the incorrect circles in red, shows the correct circle in green, and explains the rules and circle coloring in the text at the bottom. The student would then click Next and receive a randomly-generated problem of equal difficulty.



2.3.2: Add fewest and largest circles to cover all the 1s.

Circle
abc + ab'c
ac(b + b')
ac(1)
ac

✖ Use the fewest and largest possible circle(s) that contains only 1s.
Missing circle(s) shown in light green.
Incorrect circle(s) shown in red.

Table 1: Metrics for each level of the K-map activity wherein the student simplifies a randomly-generated 3-variable K-map.

Level	Avg. time spent (seconds)	Avg. number of attempts	First-time wrong %	Gave up %
1	8.5	1.4	25%	0.45%
2	4.6	1.1	8%	0.07%
3	6.3	1.1	6%	0.04%
4	9.7	1.6	24%	0.11%
5	6.9	1.1	8%	0.02%
6	12.6	1.9	45%	0.04%
7	7.5	1.1	12%	0.00%

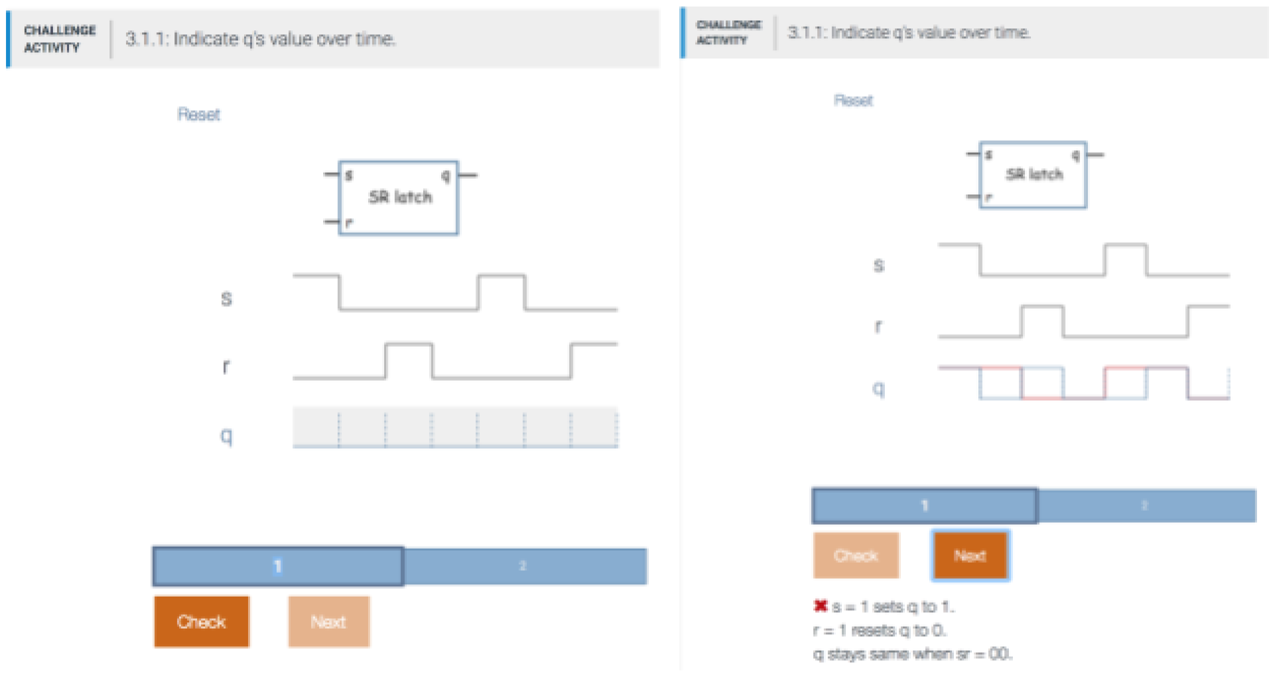
The objective of the activity was for students to demonstrate the ability to circle three contiguous 1s. The first level was much simpler than the objective, and that's by design. The first level gives the students a chance to learn the tool. Successive levels become incrementally more challenging, which likely supported the student to succeed in fewer attempts and less time. As shown in Table 1, most students correctly solved the first through fifth levels on the first try. Many students on the sixth level made one or two mistakes before correctly solving; however, most such students solved the sixth level in under two minutes. Most students correctly solved the seventh level on the first try, suggesting they had learned the objective. We took a look at submissions by students who made many attempts. One such student needed 4 tries to complete level 1, 2 tries for level 2, 1 try for level 3, 4 tries for level 4, 1 try for level 5, 10 tries for level 6, and 1 try for level 7. The student spent about 5 minutes in total. Two weeks later, the same student worked through the activity again, perhaps preparing for an exam, and completed in just over 1 minute and making only 3 incorrect submissions across all levels. Note: The section covering K-map has multiple challenge activities, and this is just 1 of them.

6. Challenge activity: Enter output of an SR latch given input s and r

This topic introduces students to sequential circuits, typically covered during the first half of the semester. An SR latch is the simplest circuit that stores 1-bit. A timing diagram is a common way to analyze the inputs and outputs of such a circuit. The objective of this activity is to familiarize the students with the workings of an SR latch. This is done with a timing diagram as in Figure 2. This activity has two levels of progression with equal difficulty. Each level presents a randomly-generated combinations of s and r, and the student needs to input the corresponding q for each combination of s and r, as in Figure 2(a). Clicking a square in q toggles between 1 and 0 values. When a student submits, the activity compares the student's q values to the expected q values. Figure 2(b) has an example of a wrong submission, including an explanation.

As shown in Table 2, many students took a few tries to solve level 1, on average 3.2 tries, then tended to quickly solve level 2, on average 1.4 tries. One interesting case was a student who spent 8.5 minutes on the first level, then 15 seconds on the second level; clearly the student learned from their experiences in level 1. The student returned 1 week later and solved both levels in 3 minutes. Then, the student returned three days later and solved both levels in 1.5 minutes, then returned 6 weeks months later and solved both levels in under a minute. The activity helps learning by enabling a student to try many input permutations via randomization.

Figure 2: (a) Student shown timing diagram for inputs s and r, then enters output q. (b) Student's submission is incorrect; activity shows correct values for q and explains how an SR latch works.



✖ s = 1 sets q to 1.
r = 1 resets q to 0.
q stays same when sr = 00.

Often, the students are comfortable with how gates work due to covering combinational circuits before this topic, but the introduction of a *latch* (or storage) is different, and takes some practice before the students can master the concept. The numbers in Table 2 clearly reflect this. As shown, the first-time wrong percentage is 74% for level 1, but drops to 17% for level 2. Similarly, the gave up percentage is 3.92% for level one but reduces to 0.21% for level 2.

Table 2: Metrics for both levels are shown, indicating students tended to require a few tries before completing level 1, but many students then quickly solved level 2.

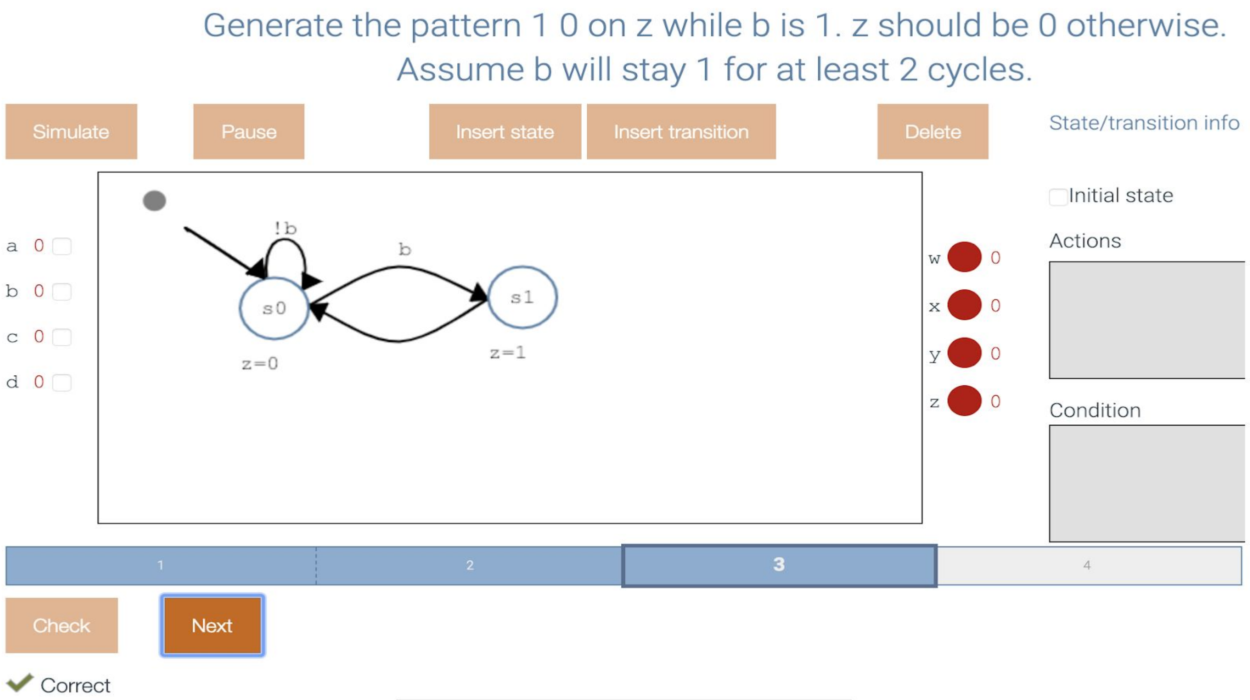
Level	Avg. time spent (seconds)	Avg. number of attempts	First-time wrong %	Gave up %
1	70.7	3.2	74%	3.92%
2	28.6	1.4	17%	0.21%

7. Challenge activity: Capture behavior as a Finite State Machine (FSM)

An FSM is a computational model capable of describing sequential behavior. This topic is typically introduced toward the end of the first half of the semester. Learning to build FSMs are a crucial but difficult aspect of any Digital Design course. The activity has 4 levels. The first level is to generate a pattern of three values to an output, such as 0 1 0. The second level is to generate a pattern of three paired values on two outputs, such as 11 01 10. The third level is to generate a toggling pattern on an output when an input is 1; otherwise, the output should be 0. The fourth level is to set the output to 1 for a cycle when an input rises; otherwise, the output should be 0.

The student is given a randomly-generated question. The student builds a state machine, initially from scratch, via the Insert state and Insert transition buttons, shown in Figure 3. The student can modify a state's actions and modify a transition's actions and condition. A transition is added by first clicking the source state then clicking the destination state. The layout of the states and transitions can be customized via drag-and-drop. A state and transition can be deleted. The student can run the state machine via the Simulate button, as well as toggle inputs.

Figure 3: FSM activity. Student is prompted with a particular behavior to capture as an FSM. Explanation has a table showing the expected and actual input and output values at each cycle.


Generate the pattern 1 0 on z while b is 1. z should be 0 otherwise.
Assume b will stay 1 for at least 2 cycles.

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

Table 3: The metrics for all four levels for this challenge activity are shown, where the gave up percentage is the highest for level 1.

Level	Avg. time spent (seconds)	Avg. number of attempts	First-time wrong %	Gave up %
1	161.2	3.7	71%	20.36%
2	136.9	1.9	32%	1.06%
3	293.4	5.8	86%	7.25%
4	219.2	4.5	78%	3.11%

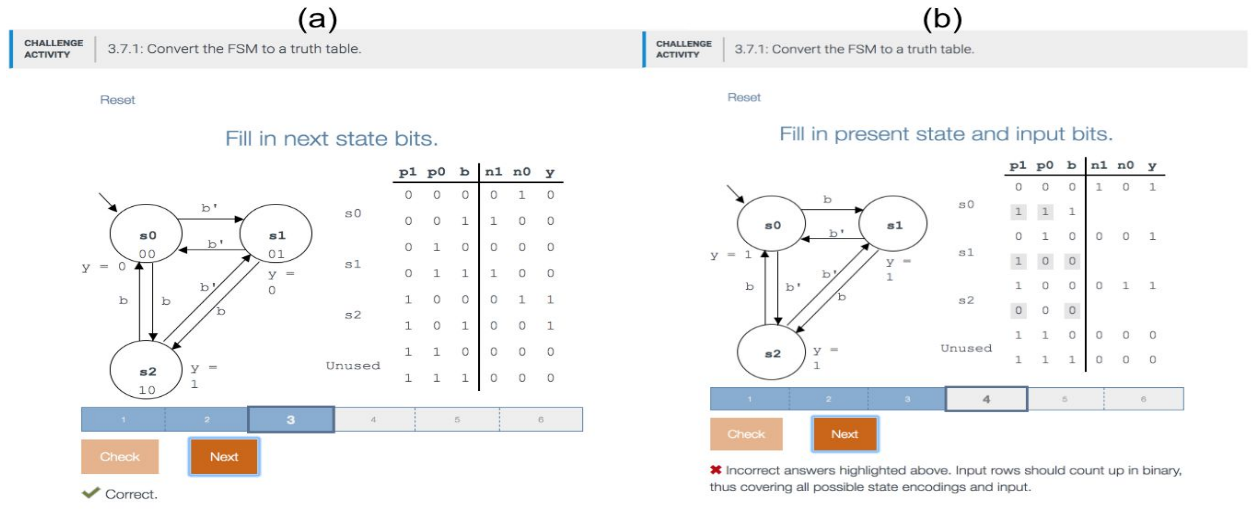
The student can click the Check button when ready for assessment. The assessment's explanation includes a table showing the expected output values and the student's state machine's output values, as shown in Figure 3. If a student's state machine was incorrect, then the student is given new randomly-generated question. Otherwise, the student progresses to the next level. Either way, the student keeps the state machine; the levels deliberately build on each other, so the prior level's state machine is typically a useful starting point for the next level's state machine. The metrics show improvement in the second level compared to the first. The third level has more attempts, which is most likely due to that level introducing an input variable in the conditions. The fourth level also has an input, but the metrics improved from the third level.

8. Convert an FSM to a truth table

The previous activity introduced students to the concept of drawing an FSM. Another important aspect is translating an FSM to a sequential circuit that consists of a register and some combinational logic. This activity helps the students learn how to convert an FSM to a truth table. The truth table derives the next state and output of the FSM based on the present state and the input. This activity has six levels. The first level has the students complete the input side of the truth table, which is entering binary values in order. The second level has the students enter the output based on the present state and input. The third level gives the students the present state, input and output, and expects the student to enter the next state. Levels 4, 5, 6 are similar to levels 1, 2, and 3 respectively. Often the task of converting an FSM to a truth table can be tedious. This activity ensures that the students gradually complete the truth table for a given FSM over three levels.

The student clicks the Start button to begin the activity. The student is presented with a randomly generated FSM with three states, one input and one output as shown in Figure 4. On the first level, the student enters the present state and clicks the Check button when ready for assessment. If the student has answered the first level correctly, the student moves on to the second level, which has the student enter the output for the same FSM. If the student entered the wrong answer, a new FSM is presented and the student can attempt the level again. On the third level, the student enters the next state for the original FSM, thus completing the truth table. Once the student has completed the first three levels, the student is presented with a new FSM on level 4, and can proceed to completing the truth table in a similar way to levels 1, 2 and 3.

Figure 4: Activity where the user converts an FSM to a truth table. (a) Correct submission for level 3. (b) Student's submission is incorrect, and the explanation states that the input rows should count up in binary.



(a) 3.7.1: Convert the FSM to a truth table.

Fill in next state bits.

	p1	p0	b	n1	n0	y
s0	0	0	0	0	1	0
	0	0	1	1	0	0
s1	0	1	0	0	0	0
	0	1	1	1	0	0
s2	1	0	0	0	1	1
	1	0	1	0	0	1
Unused	1	1	0	0	0	0
	1	1	1	0	0	0

✓ Correct.

(b) 3.7.1: Convert the FSM to a truth table.

Fill in present state and input bits.

	p1	p0	b	n1	n0	y
s0	0	0	0	1	0	1
	1	1	1			
s1	0	1	0	0	0	1
	1	0	0			
s2	1	0	0	0	1	1
	0	0	0			
Unused	1	1	0	0	0	0
	1	1	1	0	0	0

✖ Incorrect answers highlighted above. Input rows should count up in binary, thus covering all possible state encodings and input.

Table 4: Metrics for an activity wherein the student converts an FSM to a truth table.

Level	Avg. time spent (seconds)	Avg. number of attempts	First-time wrong %	Gave up %
1	44.7	2.0	40%	4.19%
2	51.6	2.5	44%	1.17%
3	74.0	3.5	41%	3.84%
4	19.5	1.4	13%	0.16%
5	36.9	1.8	24%	0.09%
6	60.2	3.3	31%	1.22%

The data shows that students spend less time on levels 1 (and 4), which is expected because students just need to enter combinations of bits by counting up in binary. Some students do not seem to make that connection quickly. We intend to expand the explanation shown in Figure 4(b) to help students more quickly make that connection. The numbers in Table 3 show that the student fared better with the second FSM (levels 4, 5, 6).

9. Conclusions

We provided detailed data for several digital design challenge activities used by thousands of students across several universities. The data shows that students are able to quickly learn the desired objectives. The auto-generation of randomized problems allowed students to repeatedly attempt problems. As found via manual inspection, some students repeatedly returned to the challenge activity days and weeks later even after correctly solving the challenge activity, demonstrating that students value the ability to return to an activity that provides a randomly-generated question. For most levels of the challenge activities, fewer than 1% of students gave up. The analysis shows that students learn on the early levels, making simple mistakes, then solve the harder levels at a quicker pace. Overall, the students learn quickly using these interactive tools with immediate feedback as opposed to submitting homework on paper and receiving feedback at a later time. Not discussed in this paper, but equally important to instructors, is that the learning occurred in a full-automated manner; no human grading was necessary to be able to assign and grade the homeworks.

References

[1] Karnaugh Map - Digital Electronics Course. http://electronics-course.com/karnaugh-map. Accessed Feb. 2019.

[2] Logic circuit simplification. http://www.32x8.com/. Accessed Feb. 2019.

[3] Karnaugh-Veitch Map. http://www.mathematik.uni-marburg.de/~thormae/lectures/ti1/code/karnaughmap/. Accessed Feb. 2019.

[4] Karnaugh Map Minimizer. http://k-map.sourceforge.net/. Accessed Feb. 2019.

[5] KMAP-Karnaugh. https://play.google.com/store/apps/details?id=com.kmapquine&hl=en_US. Accessed Feb. 2019.

[6] SR NOR Latch. http://electronics-course.com/sr-nor-latch. Accessed Feb. 2019.

[7] SR Flip-Flop. http://www.falstad.com/circuit/e-nandff.html. Accessed Feb. 2019.

[8] Another Finite State Machine (FSM) Simulator. http://www.cs.binghamton.edu/~software/fsm/fsmdoc.html. Accessed Feb. 2019.

[9] JFLAP. http://www.jflap.org/. Accessed Feb. 2019.

[10] SMCube. http://erika.tuxfamily.org/drupal/sm-cube.html. Accessed Feb. 2019.

[11] Hamada, M. and Sato, S. A game-based learning system for theory of computation using Lego NXT Robot. Procedia Computer Science, 4, pp.1944-1952, 2011.

[12] White, T.M. and Way, T.P.. jfast: A java finite automata simulator. In ACM SIGCSE Bulletin (Vol. 38, No. 1, pp. 384-388). ACM, March, 2006.

[13] Ponts, D., and Giuliano Donzellini. A simulator to train for finite state machine design. In Frontiers in Education Conference. FIE'96. 26th Annual Conference., Proceedings of, vol. 2, pp. 725-729. IEEE, 1996.

[14] zyBooks. https://www.zybooks.com/. Accessed Feb. 2019.

[15] Vahid F., Edgcomb A., Lysecky S., and Lysecky R. New web-based interactive learning material for digital design. In ASEE Annual Conference and Exposition, Conference Proceedings, June, 2016.