Interactive Asynchronous Online Computer Architecture Education

Ellen Spertus
Mills College
Oakland, California, USA
ellen.spertus@gmail.com

ABSTRACT

The COVID-19 pandemic necessitated courses being moved online and preferably made asynchronous. This was particularly challenging for small liberal arts colleges, whose faculty and students are used to close interaction. This paper describes the set of interactive asynchronous mini-lectures and online lab assignments used for an undergraduate computer architecture course at Mills College. The materials, which follow best practices for active learning, are available online for faculty at other institutions to use and modify under a Creative Commons license. We also discuss the pros and cons of making the course self-paced.

CCS CONCEPTS

• **Applied computing → E-learning**; *Interactive learning environments*; • **Social and professional topics → Computer science education**.

KEYWORDS

online education, computer architecture education, active learning, Articulate 360, Storyline 360, CircuitVerse, EDA Playground

ACM Reference Format:
Ellen Spertus. 2021. Interactive Asynchronous Online Computer Architecture Education. In *Proceedings of Workshop on Computer Architecture Education 2021 (WCAE '21).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In March 2020, faculty around the world had to move courses online because of the COVID-19 pandemic. At many schools, most classes were held online for the 2020-2021 academic year and may remain online through fall 2021. While difficult for everyone, a particular challenge for small liberal arts colleges was maintaining the expected close connection between faculty and students. Merely putting prerecorded PowerPoint lectures online would not be acceptable.

The undergraduate Computer Architecture course at Mills, a minority-serving women's college in Oakland, California, is usually taught with two weekly 75-minute lecture/recitation sections with 10-20 students and TA-staffed hours to support hardware and software laboratory assignments. We spent the summer and fall of 2020 moving the course online and would like to share the materials we developed and what we learned with faculty at other institutions.

The pandemic not only prevented students from gathering for classes but also resulted in many students moving from campus to their family homes, where they lacked the privacy or reliable internet access required for synchronous remote education. Hence, an asynchronous solution was needed, and we decided to make the class self-paced. Similarly, students were unable to access computer labs where there was necessary hardware, computers capable of running CAD software, and System-on-a-Chip boards for Verilog assignments, requiring the lab assignments to be rewritten. Despite the difficulties, we were committed to maintaining both rigor and interaction.

The next sections of this paper describe the interactive lessons that replaced lectures (Section 2), the new lab assignments (Section 3), and our decision to make the course self-paced, with no late penalties (Section 4). Grading and testing are discussed are discussed in a separate paper [5]. The final section of this paper presents our conclusions and invites others to make use of or build on our work.

2 MINI-LECTURES

STEM teaching specialist Cynthia Brame recommends these strategies to promote active learning in video lectures [1]:

- Packaging video with interactive questions.
- Using interactive features that give students control.
- Using guiding questions.
- Making video part of a larger homework assignment.

We applied all of these strategies.

2.1 E-Learning Suite Choice

The campus did not provide support for online learning beyond Canvas, the learning management system already in use, and reimbursement for equipment such as video cameras and microphones.

We were new to online education and began by evaluating e-learning software, rejecting some systems due to their high price (iSpring Suite and Elucidat), inability to import PowerPoint (Koantic), or inability to create interactive truth tables (Adobe Captivate). Our final choice was Articulate 360 due to its functionality, documentation, and active support forums. Costing $499 for a one-year license, it was by far the most expensive part of the course and the only software that we paid for. It is worth noting that a free 30-day trial is available, making it possible for an instructor to modify our materials for their course without buying a subscription.

Presentations created by Articulate 360 can be viewed without a license.

Articulate 360 is a suite consisting of a number of programs and services. The ones we used were:

- Storyline 360 for creating interactive e-learning lessons
- Replay 360 for creating (non-interactive) instructional videos
- Review 360 for hosting the lessons generated by Storyline 360

Of these, only Storyline 360 was essential. We could have created instructional videos for free with Screencast-O-Matic and hosted our generated lessons on any web server.

2.2 Storyline 360

Storyline 360 creates "stories", which we refer to as "mini-lectures", out of sequences of slides, which can accept input and run arbitrary calculations to determine what is displayed next. An example of a non-interactive slide is shown in Figure 1. The top portion shows all of the elements that students will see and hear, including a video of the instructor. The bottom portion shows a timeline that controls when elements appear or disappear. When the slide starts, some visual elements appear and a video of the instructor begins playing when the slide starts. The textboxes with the labels "iOS" and "OS X" appear at specified later times in sync with what the instructor is saying. Instructors wishing to adapt the slides for their own classes could re-record the video and manually change the start times for the labels to match their delivery.

If this were all Storyline 360 could do, it would be no better than PowerPoint. The next section describes the interactive features.

2.3 Interactive Mini-Lectures

Although we did not read Brame [1] until preparing this paper, we found that our practices followed her recommendations for student engagement:

(1) Keep each video brief.
(2) Use conversational language.
(3) Speak relatively quickly and with enthusiasm.[1]

Specifically, she wrote that these strategies would increase the percentage of each video that students watch, decrease mind wandering, and create "a sense of social partnership between student and instructor".

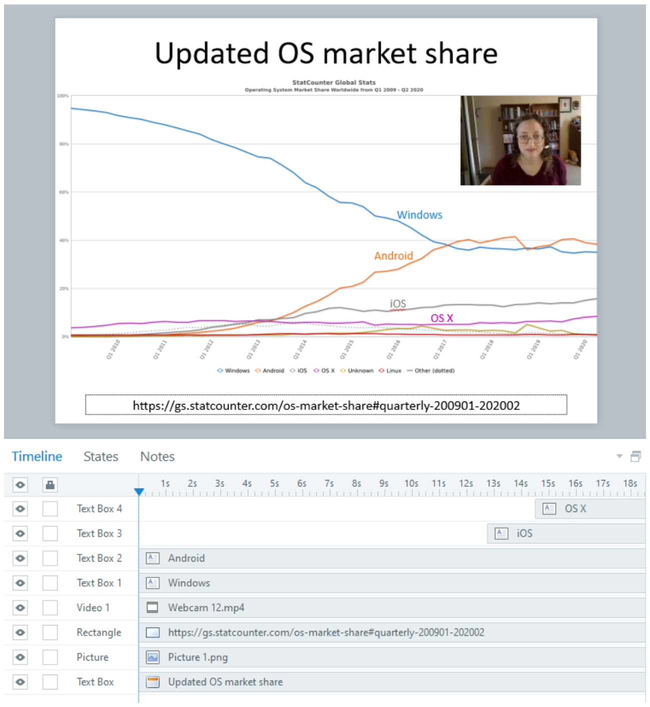
2.3.1 *Embeddings.*

Almost all slides contained a video of the instructor, which began playing when the slide appeared, and the slide advanced when the video completed.

The simplest interactive slides were ones that let the student control the playing of additional audio and video, such as when the instructor introduced YouTube video clips of Danny Hillis or a biography of George Boole. For security reasons, the user needed to click play to start externally hosted videos.

A slightly more interesting example was when the instructor introduced the term "algebra" and gave its etymology, the title of the book *The Compendious Book on Calculation by Completion and Balancing*, whose original (long) Arabic name contains "al-ǧabr". After the instructor explained this and played a recording we commissioned of an Arabic speaker reading the title, a "replay" button appeared on the screen, enabling the student to replay the audio as many times as desired. This was implemented by making a button appear after the recording was played for the first time and adding the "trigger":

```
Play audio PronunciationAudioClip2
    When the user clicks Button1
```



Updated OS market share
StatCounter Global Stats
Operating System Market Share Worldwide from Q1 2009 - Q2 2020
Windows
Android
iOS
OS X
https://gs.statcounter.com/os-market-share#quarterly-200901-202002
Timeline
States
Notes

Text Box 4	OS X
Text Box 3	iOS
Text Box 2	Android
Text Box 1	Windows
Video 1	Webcam 12.mp4
Rectangle	https://gs.statcounter.com/os-market-share#quarterly-200901-202002
Picture	Picture 1.png
Text Box	Updated OS market share

Figure 1: A partial screenshot of Storyline 360 showing a slide and the associated timeline. Most of the elements appear when the slide starts, but the text boxes containing "iOS" and "OS X" appear approximately 12.75 and 14.5 seconds after the start.

An example of a more complex embedding is shown in Figure 2, which introduces `not` gates and embeds interactive logic from CircuitVerse, a free cloud-based logic simulator. If the student clicks on the 1 input to the inverter, the input is toggled to 0 and the output changes to 1.

Figure 3 shows a 4-bit ripple carry built from full adders. In addition to toggling the inputs, the student can switch the view to the implementations of the full adder and the half adder out of which the full adder is built.

2.3.2 *User Input.*

In addition to containing a simulation, Figure 2 contains input fields, initially containing question marks. As explained in the video that is played on the slide, students must replace the question marks with the correct values and click the check mark on the bottom right to advance to the next slide. If a student gets

[1] A downside of speaking quickly is that it could make speech hard to understand for students for whom English isn't their first language. This could be addressed with closed captions, which can be manually added in Storyline 360 but are not generated automatically.

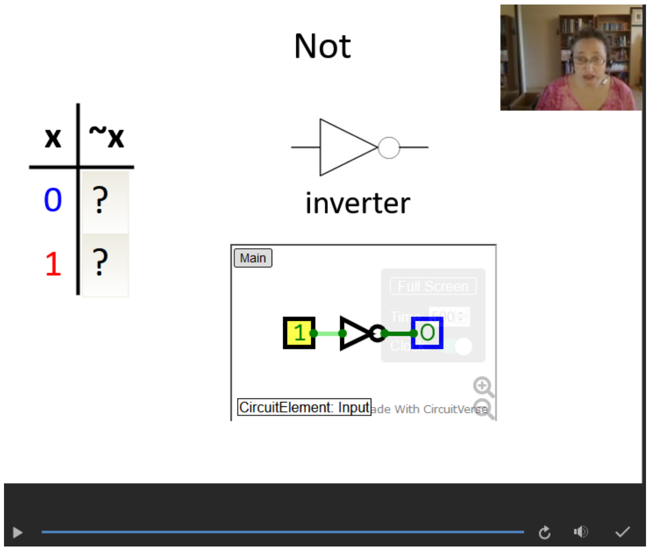


Figure 2: A slide as seen in a web browser with input fields and a live `not` gate, embedded from the website CircuitVerse.org.

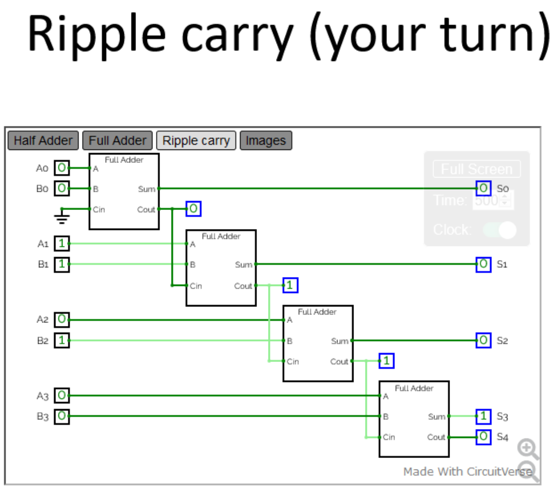


Figure 3: A ripple carry composed of full adders, composed of half adders. Students could view and toggle the inputs of any of these elements.

a question wrong, they receive visual or audio feedback and are encouraged to try again.

Other slides combine button-controlled media with user input, such as a slide showing how to convert from base-two to base-ten, then asking the user to do a conversion (Figure 4). In addition to displaying input fields, a button labeled "HELP!!!" appears for students unable to answer the question. It introduces a YouTube video demonstrating (in a different way) how to solve the problem. Hint buttons in other mini-lectures navigate the student to otherwise hidden slides.

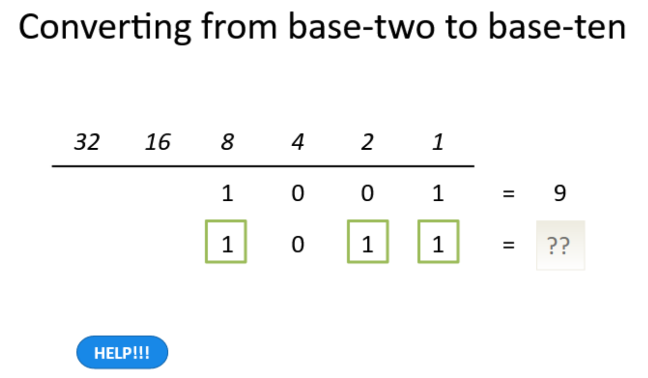


Figure 4: A slide in which the user can enter input in the gray box or request help.

Different types of user input are possible. Figure 5 shows a sequence of slides used in teaching Karnaugh Maps. In the first step (a), the student must fill in each text box with the appropriate values for a given truth table. If they enter a wrong value, they receive immediate audio feedback. Next (b), the student must drag a shape over the appropriate portion of the Map. Finally (c), the student must click on a button to select the right formula. For each incorrect answer, an audio clip is played. For example, if they select `A` or `~A`, they hear: "No, the formula wouldn't include `A` because `A` is sometimes 0, sometimes 1 in the region." If they select `~B`, they hear: "No, `~B` would be correct if `B` were always 0 in the red region, but `B` is always 1." Additional slides, not shown, have the user select the remaining region and enter its formula before providing the full formula for the Map.

In another Karnaugh Map problem where the user types in a formula for a region (`~C`), we used JavaScript to select the appropriate audio feedback.

```
var player = GetPlayer();
var guess = player.GetVar("TextEntry19").trim();
if (guess === "C" || guess === "c") {
    player.SetVar("Message", "C");
}
else if (guess.includes("A") || guess.includes("a")) {
    player.SetVar("Message", "A");
}
else if (guess.includes("B") || guess.includes("b")) {
    player.SetVar("Message", "B");
}
else if (guess.includes("~")) {
    player.SetVar("Message", "Not");
}
else if (guess.includes("&")) {
    player.SetVar("Message", "And");
}
else if (guess.includes("|")) {
    player.SetVar("Message", "Or");
}
else {
    player.SetVar("Message", "Other");
}
```

Separate triggers played different audio clips depending on the value that `Message` was set to. For example, if their formula was "C" or "c" (correct), they heard: "Woo hoo! You got it! The answer is `C`. Choosing the largest region gave you a very simple formula" and the slide advanced. If their formula had unneeded terms, operators,

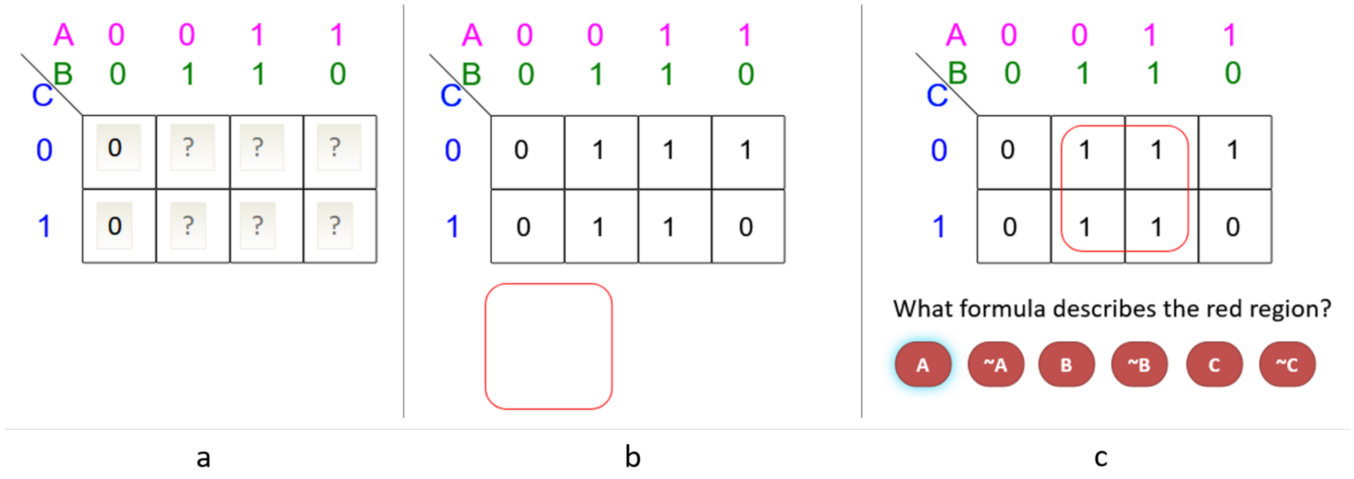


Figure 5: A sequence of slides in which the student (a) fills in a Karnaugh Map for a given truth table, (b) drags a shape over the largest region, and (c) selects the formula for the region.

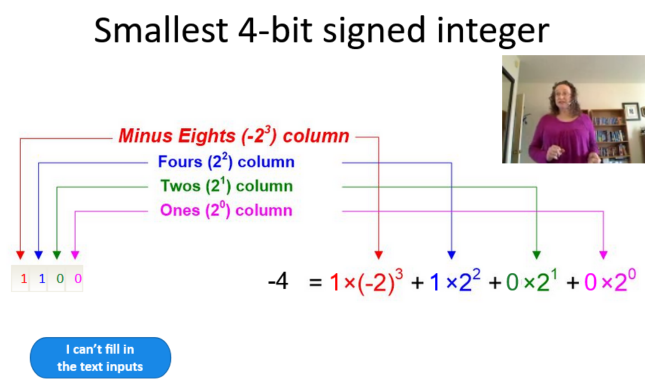


Figure 6: A slide in which the bits entered by the user on the left affect the corresponding coefficients on the right half of the slide and the result (currently -4) in the center of the slide.

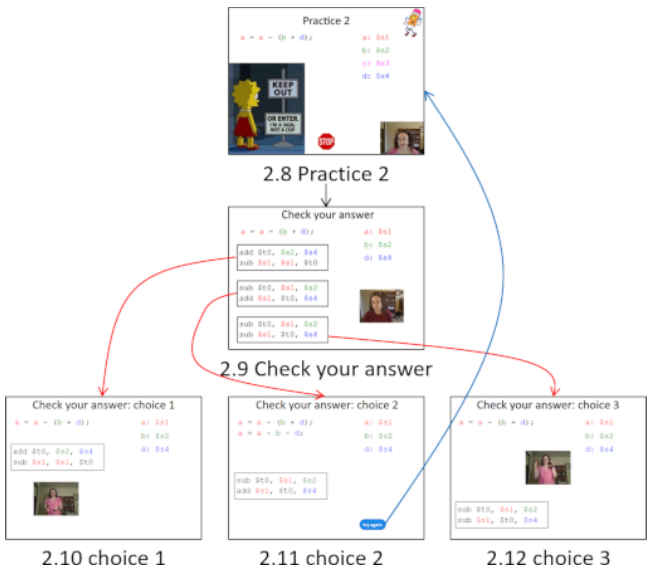


Figure 7: Control flow of slides, depending on user behavior.

or other characters, they received a tailored message and were encouraged to try again or ask the instructor for help.

In the mini-lecture introducing two's complement notation, students are asked to determine the smallest (most negative) number than can be expressed in 4 bits. As they enter 1s and 0s in the four bit positions, the corresponding coefficients on the right and the sum in the center are updated. When they click the submit button (not shown), they are told if they got the correct answer (-8) or invited to try again with an appropriate message if their answer was incorrect. If the user lacks the fine motor ability to enter the bits, they can press the blue button to get a multiple-choice version of the problem.

Figure 7 shows how a student's actions affect which slides are shown. Slide 2.8 asks the student to write some assembly code before advancing to the next slide (2.9), which instructs the student to click on one of 3 possible answers (2 of which are correct) or to contact the instructor if their answer is not shown. Their choice determines which of 3 different slides they see next. If the student chooses the wrong answer, slide 2.11 is shown, which explains why the answer is incorrect and invites them to try again. If the student clicks the blue button, which is labeled "try again", slide 2.8 is again displayed. Slides 2.10 and 2.12 are shown in response to correct answers.

2.3.3 Survey Results. We asked all 13 students who began the course to complete a survey evaluating different aspects of it. Ten of the students completed the survey. As shown in Table 1, only one student disagreed with the statements "I enjoyed the mini-lectures" and "Answering automatically-graded questions was helpful"; the rest responded positively or neutrally. Most students found it helpful to be able to stop and replay the material but did not feel doing so was more time-consuming than regular lectures. We were concerned that the mini-lectures might be buggy, since they had to run on several different desktop and mobile browsers and we were inexperienced with the software, but only 3 students agreed the slides were buggy. Two of these students tended to work ahead of their classmates and reported bugs, which we fixed, giving the slower students a better experience.

2.4 Pre- and Post-Lesson Questions

Brame writes that providing guiding questions for videos "may increase germane cognitive load, improve memory via the testing effect, and improve student self-assessment" [1]. Darby and Lang agree that "[a]n effective and simple way to create ... engagement and add accountability is to include a short, graded assessment after each required video" [2].

Before opening each mini-lecture, students were shown related questions that they needed to answer afterwards (or occasionally beforehand). For example, before the first lecture, "What is a Computer?", students were instructed to take a few minutes to think about and give a definition for a computer, without looking anything up. Writing prompts for after the lecture were:

(1) What was your first computer, and how did you use it?
(2) If microprocessors keep getting smaller and cheaper, how might they be used 20 years from now? [This alluded to a story told by Danny Hillis in an embedded video.]
(3) Is there anyone or anything from the history of computers that you wish I'd mentioned?
(4) Was any part of the lecture unclear?

Some later lectures had short technical questions, such as asking a student to find a formula for a control bit or convert a snippet of Java code into assembly language. These allowed the student to immediately practice what they learned, replaying parts of the lesson if necessary, and get feedback within a few days from the instructor.

Other questions tied the material to the student's life. For example, after a lesson on metric prefixes and the history of hard drives, students were asked:

- What's the largest metric unit you've had cause to use outside of school? What was the smallest?
- Have you bought storage recently? How much did you pay, in what year, for how much memory? How did you feel about the price? Were you aware how prices had fallen?
- Do you still run out of disk space?

Students were also invited to ask questions about anything they found unclear.

Although the questions did not count toward the semester grade, students completed them. The instructor answered any questions they contained (or asked students to meet with her for an explanation) and left comments on the responses. She felt this helped her get to know the students and give them personalized attention. As Table 2 shows, students generally thought completing the writing prompts was worth the time it took and that it enabled them and the instructor to get to know each other.

One student wrote:

> Sometimes when people talk about reading a book they mention the issue of "I read 2 pages and did not sustain any of that information" So without the prompt questions I would've watched 4 lectures and not had a stop and check in with myself.

2.5 Live Sessions

There were two lectures that we felt would be difficult to present asynchronously. Since the class was self-paced, the instructor met with students synchronously online in groups of 2–3 for these topics. The first was translating loops into assembly language, where the instructor presented and explained examples and then had students create flowcharts and assembly code, which they reviewed together. This also gave her the opportunity to clear up confusion on earlier assembly programming topics, such as procedure calls.

The other synchronous session was for introducing the simple microprocessor that was the subject of the last lab assignment and a final exam question. While it could have been presented asynchronously, so few students (6) were still in the course by the end of the semester (a point discussed below) that talking through the material in small groups was a better use of instructor time than creating a mini-lecture, and this allowed the instructor to give immediate feedback on a traditionally in-class exercise to determine the control bits for the instruction set and to write a factorial function for the microprocessor.

Although the survey did not ask about the live sessions, one student volunteered that they "were helpful because they were in a small group of people and everyone could ask a question if they needed to and that was really nice."

3 LAB ASSIGNMENTS

The ordinary version of the class has 6 lab assignments, all of which were replicated in the online version.

3.1 TTL Lab

The first lab assignment is to design and build a full adder using TTL components. Ordinarily, students use semi-portable digital trainers weighing approximately 20 pounds each. While we still made these available to students living on or near campus, we provided inexpensive alternatives that we shipped to students not living nearby. In addition to providing each student with the ordinary tools and supplies (multiple colors of wire, wire strippers, plastic logic templates, chip pullers, pliers, TTL chips, and logic probes), we provided non-powered breadboards, discrete LEDs, and 1 KΩ current-limiting resistors from our inventory. To power the breadboards, we bought MB102 breadboard power supplies with 9V battery clips (Figure 8), which were surprisingly inexpensive (4 sets for $8.99 from Amazon) but had a high failure rate, requiring us to test each set before sending it out and either allowing time to receive replacements or ordering twice as many as were needed. While the listing said that the MB102 could be powered through a USB cable, reviews indicated that this was not the case, so we used 9V batteries.

We shipped the supplies in USPS Priority Mail flat-rate boxes ($15.05 each), which was generally fast, except for one shipment

Table 1: Survey respondents' level of agreement with statements about the mini-lectures.

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I enjoyed the mini-lectures.	4	3	2	1	0
Answering automatically-graded questions was helpful.	3	4	2	1	0
Being able to stop and replay material helped me learn the material.	7	1	1	1	0
The mini-lectures took more time than regular lectures would have.	0	2	4	1	3
The mini-lectures were buggy.	0	2	4	1	3

Table 2: Survey respondents' level of agreement with statements about pre- and post-lecture questions.

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Completing the writing prompts was a waste of time.	0	1	5	2	2
I felt I got to know Ellen through the lessons and comments on my responses.	3	4	2	1	0
I felt Ellen got to know me through my responses.	1	6	3	0	0

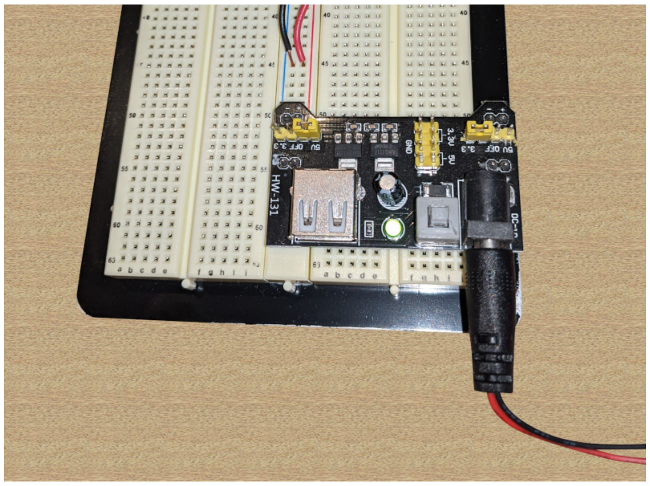


Figure 8: A breadboard with an MB102 power supply connected to a 9V battery, not shown.

from San Francisco to nearby Oakland, which traveled unnecessarily across the country and back over a period of more than a week. Using flat-rate postage enabled us to purchase and email return postage for the students to print and use when returning the supplies.

Instead of teaching students how to use the equipment and helping them debug their circuits in person, we had to do both remotely (via Zoom), which worked surprisingly well.

3.2 Simulation/CAD Labs

We have one assembly language lab and 4 CAD labs. We use MARS for assembly language simulation, which was unchanged when we moved online. Traditionally, we use Altera Quartus Prime Lite for the CAD labs, beginning with schematic capture and culminating in burning Verilog code onto Terasic DE1-SoC boards. Because the software is so complicated, the assignments are long and require much TA support.

For the online course, we chose a cloud-based solution because (1) some of our students did not have powerful computers at home and (2) we did not need the ability to program FPGAs. We chose EDA Playground, a free cloud-based solution.

In normal years, the first Verilog assignment includes working through a 41-page Quartus Prime tutorial and then using schematic capture to simulate the full adder they implemented in the hardware lab. Because EDA Playground is so simple, there was no need for a tutorial. Students were given a Verilog implementation and test bench for a half adder, which were explained in a video, and were required to extend it to a full adder.

The previous time the instructor taught the course with Quartus Prime, students reported spending between 1:45 and 7 hours on the assignment, with an average of 3:35. With EDA Playground, students reported spending between 1:30 and 2 hours, with an average of 1:50.

The remaining CAD labs covered exactly the same Verilog as in the ordinary course, requiring students to:

- Add overflow detection to the MIPS ALU provided in the textbook (to begin learning Verilog and improve their understanding of two's complement).
- Implement a 4-way multiplexer out of 3-way multiplexers (to understand how more complex elements could be built from simpler ones and begin creating Verilog modules).
- Complete the implementation of a microprocessor by creating the control unit and the top-level module that instantiated and connected the other modules. Students were given a factorial program for the processor and required to write a fibonacci program.

The process went so much more smoothly in EDA Playground than in Quartus Prime that it will be tempting to keep using the former, although that would deprive the students of the opportunity to burn the Verilog onto hardware. We might make the use of

Table 3: Survey respondents' level of agreement with statements about the course being self-paced.

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The lack of due dates...					
...lowered my stress level.	6	1	1	1	1
...gave me flexibility when I needed it.	5	2	1	2	0
...caused me to prioritize other classes over this one.	3	4	2	1	0
...led me to procrastinate.	2	3	2	2	1
...led me to spend more time on assignments.	4	2	3	1	0
...led to my having to withdraw from the course.	1	0	1	4	4

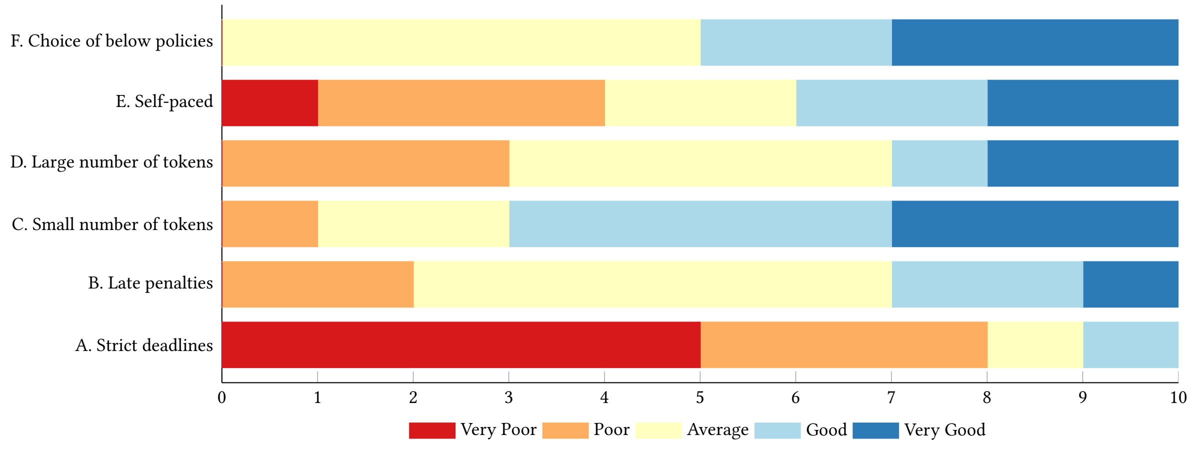


Figure 9: Ratings of Different Late Assignment Policies

Quartus Prime a challenge assignment for students wishing to earn the highest grades or leave it for a classroom demonstration.

4 LATE ASSIGNMENT POLICIES

As alluded to above, the course had a high attrition rate. Of the 13 students who began the course, 5 completed it on time, 1 completed it a month late, and 7 withdrew from the course, as late as the last day of class, which was permitted under special pandemic rules. Typically, attrition rates are low (no more than 10%) for upper-division courses, so this was extraordinary. It is unclear how much was due to each of the following factors:

(1) students' external issues (such as mental and physical illness and poor Internet access)
(2) the asynchronous format
(3) the course being self-paced

While students were given recommended due dates for each assignment (including mini-lectures), they were told there would be no late penalties, and the scheduled class time (two 75-minute sessions per week) was used as office hours. Perhaps predictably, almost all students quickly fell behind. In an attempt to keep students on track, we announced a few weeks into the semester that attendance of class Zoom sessions was now mandatory, during which they could work at their own pace, with the instructor available to answer questions. Despite this change in policy and students' perennial optimism about their ability to catch up, most students were unable to do so.

As shown in Table 3, students generally agreed that the lack of strict due dates lowered their stress levels and gave them needed flexibility, but many acknowledged that it caused them to prioritize other classes, to procrastinate, and to spend more time on assignments (some productively, some due to perfectionism). With the exception of one student, they did not, however, blame their attrition on the policy.

Figure 9 shows students' ratings of different hypothetical late assignment policies, with more lenient policies higher up the y-axis. The bottom row shows that students rated policy A, strict deadlines (with no late assignments accepted), as very poor (5), poor (4), average (1), or good (1); none rated it a very good policy. Students much preferred policy B where late assignments were accepted with a per day penalty; all students except for one rated it higher than not accepting late assignments. Policies C and D involved tokens that a student could use to turn in an assignment one day late without penalty. All students except for (the same) one preferred having a small number of tokens (C) over having no tokens (B) or rated the options equally. Perversely, having a small number of tokens (C) was generally preferred over having a large number of tokens (D), where the boundary was 8 tokens. Policy E was the one used in the course, self-paced with no late penalties. While we would have predicted that students who passed the course would have rated it highly and those who did not would not, there was no clear correlation. (Two of the 4 students who rated it "poor" or "very poor" passed the course, as did 1 of the 2 who rated it "average" and 3 of the 4 who rated it "good" or "very good".) Option F was to let students choose among the policies, which none of the students rated negatively.

One student who did not complete the course commented:

> I think that there needs to be some sort of structure and accountability system. Right now people have a lot of personal stuff that they are dealing with and when you have really flexible late policies it feels like you have time to put those things aside to maybe focus on something more urgent, but then things get away from you.

Another student who did not complete the course and volunteered to the teacher that they had ADHD, wrote:

> [L]ate assignment penalty can be ablest[2] to some students. But I also think no deadlines/due dates can lead to unproductive perfectionism which I have experienced.

A student who completed the course wrote:

> [B]eing able to go at my own pace really helped me. Towards the beginning of the semester, when there wasn't much happening in my other classes, I was able to get ahead, so that once my other classes picked up I could dedicate more time to those.

Clearly, there is no one best policy. We are still debating what to do when the course is next offered.

5 CONCLUSION

We used free and commercial software to move an introductory undergraduate computer architecture course online during the pandemic, following best practices for active learning, such as creating short interactive videos packaged with guiding questions. Students largely enjoyed the mini-lectures and felt their learning was facilitated by the automatically-graded questions and the ability to replay portions. Despite the asynchronous format, they and the instructor felt they were able to get to know each other, which is helpful for advising, mentoring, and retention [3].

The course had high attrition. We do not know how much was due to the format, the decision to make it self-paced, and students' external challenges. We suspect it was an interaction among those factors: students living off campus in situations not conducive to learning prioritized other courses with strict deadlines over our course and eventually fell too far behind to recover. When asked what policies they advised for future semesters, students preferred policies with some leniency over the extremes of no deadlines or no late assignments accepted.

We hope that the materials we created can be useful to others. We invite other computer architecture instructors to adapt and reuse our material, either as primary or supplementary material, and we have done a soft launch of a free non-credit Canvas course. Access to both can be obtained at the author's website [4].

[2]The term "ableist" refers to prejudice or discrimination against people with disabilities.

REFERENCES

[1] Cynthia J. Brame. 2016. Effective Educational Videos: Principles and Guidelines for Maximizing Student Learning from Video Content. *CBE—Life Science Education* 15, 4 (Winter 2016), 1–6. https://doi.org/10.1187/cbe.16-03-0125 Published online 13 October 2017.

[2] Flower Darby and James M. Lang. 2019. *Small Teaching Online: Applying Learning Science in Online Classes.* Jossey-Bass, San Francisco, CA.

[3] Ernest T. Pascarella. 1980. Student-Faculty Informal Contact and College Outcomes. *Review of Educational Research* 50, 4 (1980), 545–595. https://doi.org/10.3102/00346543050004545

[4] Ellen Spertus. 2021. Computer Architecture Teaching Resources. https://ellenspertus.com/comparch-resources.

[5] Ellen Spertus and Zachary Kurmas. 2021. Mastery-Based Learning in Undergraduate Computer Architecture. In *WCAE '21: Proceedings of the 2021 Workshop on Computer Architecture Education.* ACM, Valencia, Spain.