Implementation of a Flipped Classroom Model in Digital Logic Design

Introduction

Studies repeatedly show improvements in learning, achievement, and success for students after implementation of active-learning and student centered teaching practices. Active learning improves retention of content, achievement level, and success in courses (Felder, Woods et al. 2000, Freeman, Eddy et al. 2014). In a meta-analysis of 225 studies on active learning in STEM, significant improvements in achievement and in DFW rates were seen across all STEM disciplines, course levels, and course sizes (Freeman, Eddy et al. 2014) after the introduction of active learning. Cooperative learning is recommended to be used with active learning activities, and improves content retention, critical thinking and problem solving skills (Felder, Woods et al. 2000). This paper describes the transformation of a freshman level Digital Logic Design course to a fully flipped model, and the impacts on learning seen after the first iteration of the change.

Introduction to Digital Logic Design is an introductory course in digital logic circuits covering number representation, digital codes, Boolean algebra, combinatorial logic design, sequential logic design, and programmable logic devices. It is a four hour course with a required lab and optional discussion section. It is offered in the Fall and Spring semesters and required for Electrical Engineer (EE), Computer Engineer (CoE), and Computer Science (CS) majors. Two sections of the course are taught each semester, typically by the same instructor. It is a gateway

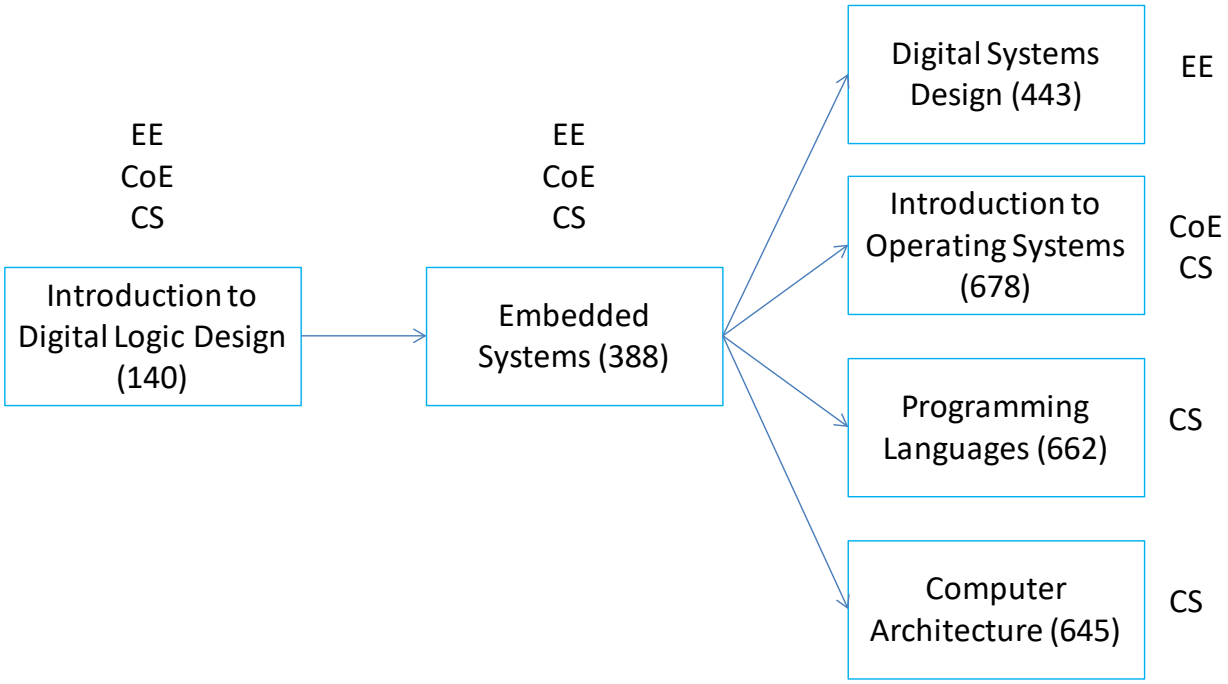


Figure 1. Curriculum Map for Digital Logic Design

course to several other required courses in the department as illustrated in Figure 1. Approximately 110 students take the course each semester. The course includes honors students who are assigned additional homework and lab assignments.

On completion of the course, the student is expected to be capable of: 1) representing a combinational logic function as a truth table, Boolean expressions including various canonical forms, and logic circuits, and translating between these representations; 2) translating a simple logic problem expressed in prose to a combinational logic function; 3) simplifying a combinational logic function using K-maps and other techniques; 4) converting numbers between decimal and binary (and related) forms and designing simple digital circuits to perform numerical arithmetic functions; 5) designing combinational circuits using common building blocks; 6) designing flip-flop, register, and counter circuits; 7) implementing simple finite state machines from written specifications; and 8) writing VHDL code for simple digital circuits.

Introduction to Digital Logic Design has been taught at this institution for many years and is a typical introductory course in digital logic circuits which is included in most engineering curriculums. It has been taught in a variety of formats by a number of instructors. Generally, it has been taught by someone with a PhD and has not been taught by graduate teaching assistants, although graduate teaching assistants generally teach the lab sections. In this paper, we focus on two semesters taught by a single instructor. The first semester was taught utilizing a mix of lecture and active-learning activities. The second semester was taught fully flipped, with all information delivery occurring outside of class and active learning activities during the entirety of the class time. Each semester also had a lab component to the course, which was not impacted by the changes made to the "lecture" portion of the course. Each semester, the instructional team consisted of the main instructor, an undergraduate assistant, and 2-3 "undergraduate teaching fellows," who assisted with answering questions during in-class activities.

Course Implementation

Fall 2017: Hybrid lecture and active-learning

At the beginning of the semester, the students were given a schedule of the topics that would be covered each class period and the sections in the textbook corresponding to those topics. The students were not required to prepare for class as the lectures were designed to provide all the instruction the student would need to complete the homework and pass the exams. The PowerPoint slides for each lecture were posted on the University's Learning Management System (LMS) approximately a week before the lecture for students to study.

Each section of the course was held in a traditional, lecture-style classroom or auditorium. During the 75 minute class periods, the instructor lectured for approximately 30 minutes. Previously posted PowerPoint slides were presented and annotated with an electronic stylus. The lecturer stopped after each major point and asked for questions. After the lecture, the students were given an in-class problem to complete and hand-in at the end of class. The students were encouraged to work together to solve the problem, but each student had to turn in their own in-class problem. The in-class problems were graded and comprised 10% of the student's grade.

The instructor and two undergraduate teaching assistants, who had been given the solutions, walked around the room helping students as they worked on the problem. A typical in-class problem was: Design a circuit that multiplies an 8-bit unsigned number by 9, using only 1 full-adder.

Each week the students were given 8-10 problems from the textbook to complete as homework. Honors students were given 2-3 additional problems, which might require further reading from the textbook. The homework assignments were posted at least a week before they were due and at the same time the lecture slides corresponding to the assignments were posted. An optional 2-hour help session was held the night before the homework assignments were due. The help session was led by an undergraduate teaching assistant. All of the homework assignments were graded and comprised 20% of the student's grade. The solutions for the homework assignments were posted on-line shortly after the assignments were turned in. A typical homework problem might be: *(a) Show that the circuit in Figure P3.2 is functionally equivalent to the circuit in Figure P3.1. (b) How many transistors are needed to build this CMOS circuit?*

Three open-book, open-note non-comprehensive exams were given. The students were allowed to use their phones, tablets, or laptops to access their notes, lecture slides, and homework solutions. The schedule for the exams was given on the first day of class and not changed. The first two were given during the 75 minute class periods and the last one was given during the class's 150 minute final exam period. The last exam was approximately twice as long and covered twice as many topics as the first two. Honors students were given the same exam as the non-honors students. The exams were equally weighted (including the longer final exam) and represented 45% of the student's grade.

During the class period prior to the exam, the students were given the number of questions and points per question along with the learning objective that the question covered and the section of the book where the learning objective was covered. For example: Create a truth table for a logic function – Section 2.3 (6 points). An exam review was also conducted by the undergraduate teaching assistant during the help session the night before the exam. A typical exam question might be: *Create a truth table for the logic function: $f = x(x \cdot y)(y+z)$.*

Spring 2018: Fully Flipped Classroom

The second semester was taught using a flipped classroom model, and facilitated in a classroom specifically designed for active learning. Students sat in 5 person U-shaped tables, and the instructor and assistants were easily able to move around and assist students as they worked (Figure 2).Students were responsible for reviewing the material and taking an online quiz prior to class. They could review the material by listening to the instructor in pre-recorded lectures, reading the book, going over the slides, looking it up on the internet, or all of these. During the class period, they worked on in-class problems. Each in-class problem covered 1 of 84 learning objectives (e.g., what is the truth table for the AND function) and was graded.

In the second semester, the two sections of the class were held on different schedules. One section was three 50-minute class periods on Monday, Wednesday, and Friday (MWF); the

second section was two 75-minute class periods on Tuesday and Thursday (TuTh). Because of the differences in the length of the lectures, the MWF students did two in-class problems each class period, while the TuTh students did three in-class problems. So, six learning objectives were covered each week. The pre-record videos, slides, and on-line quizzes were broken down by learning objective, so the MWF students had to prepare for two of them before each class and the TuTh students had to prepare for three before each class.

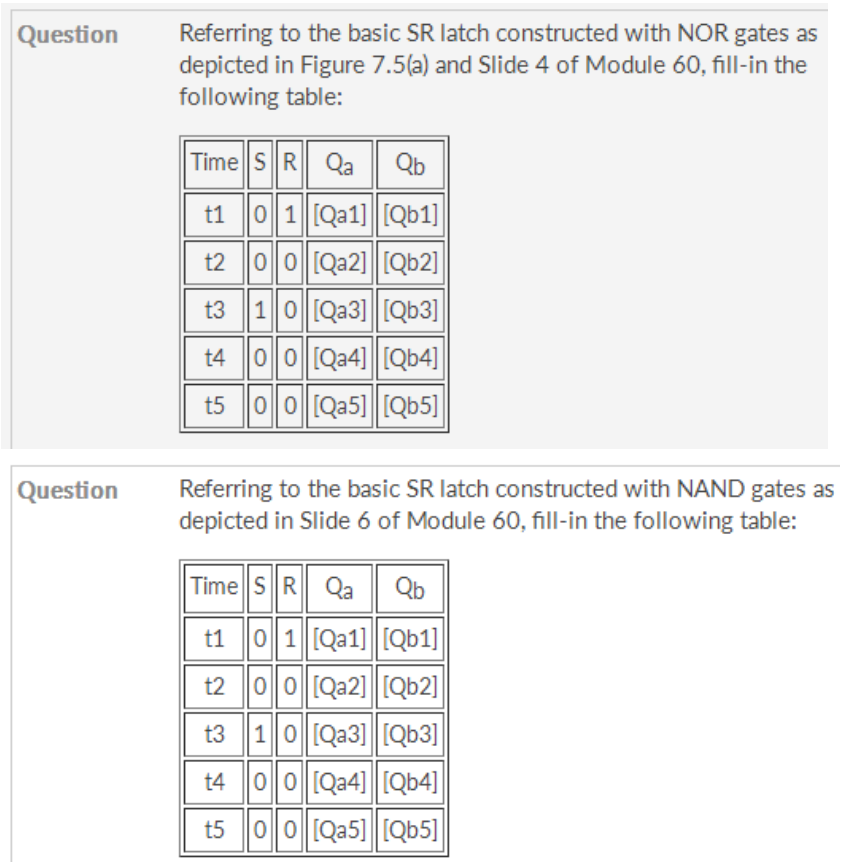
The learning objectives for the pre-flipped first semester were covered in 28 lectures which were each approximately 30 minutes in length. The goal of the flipped-class was to divide these lectures into 84 modules each representing approximately 10 minutes of the first semester lectures. The 84 learning objectives for the flipped class are shown in Table 1.

Figure 2. Photo of flipped classroom implementation

Table 1: Learning Objectives for Flipped Classroom.

Objective Number	Objective Description	Objective Number	Objective Description
1	Give the truth table for the AND and OR functions.	43	Design a carry-lookahead adder.
2	Give the truth table for the NOT of a logic function.	44	Design a hierarchial carry-lookahead adder.
3	Create a truth table for a logic function.	45	Design an array multiplier for unsigned binary numbers.
4	Draw the logic network of gates that implements a logic function.	46	Multiply signed binary numbers with 2's complement arithmetic.
5	Use Boolean Algebra to reduce a logic function.	47	Convert a fixed-point binary number to decimal.
6	Prove a Boolean identity with a Venn Diagram.	48	Give the decimal exponent range and precision of a single- or double-precision IEEE floating point number.
7	Give the canonical sum-of-products (SOP) for a logic function.	49	Design an n-to-1 multiplexer.
8	Give the canonical product-of-sums (POS) for a logic function.	50	Design a switch with a multiplexer.
9	Determine whether to use a canonical sum-of-products (SOP) or product-of-sums (POS) to implement a logic function.	51	Synthesize a logic function with a multiplexer.
10	Give the truth table for the NAND and NOR functions.	52	Factor a single variable out of a Boolean expression with Shannon's Expansion Theorem.
11	Draw the logic network of only NAND gates that implements a logic function.	53	Apply Shannon's Expansion Theorem to a single variable of a logic function to implement it with a multiplexer.
12	Draw the logic network of only NOR gates that implements a logic function.	54	Apply Shannon's Expansion Theorem to multiple variables of a logic function to implement it with a multiplexer.
13	Give the truth table for the XOR function.	55	Design a binary decoder.
14	Give the differences between a positive logic and a negative logic implementation using voltage levels.	56	Design a binary encoder.
15	Know which voltage (high or low) applied to a NMOS transistor opens or closes the transistor switch.	57	Design a priority encoder.
16	Know which voltage (high or low) applied to a PMOS transistor opens or closes the transistor switch.	58	Design a comparator for two unsigned numbers.
17	Draw the NMOS realization of a logic network.	59	Design a comparator for two signed numbers.
18	Draw the CMOS realization of a logic network.	60	Design a basic SR latch with NOR and NAND gates.
19	Use a Karnaugh map to find the minimum-cost sum-of-products (SOP) for a 2-variable logic function.	61	Design a gated SR latch with NAND gates.
20	Use a Karnaugh map to find the minimum-cost sum-of-products (SOP) for a 3-variable logic function.	62	Design a gated D latch.
21	Use a Karnaugh map to find the minimum-cost sum-of-products (SOP) for a 4 and 5 variable logic function.	63	Design a master-slave D flip-flop with two gated D latches.
22	Identify the literals, implicants, prime implicants, cover, and cost of a logic function.	64	Design a positive edge triggered D flip-flop from NAND gates.
23	Know how to use essential and non-essential prime implicants to find a minimum cost cover for a sum-of-products (SOP) implementation.	65	Design a negative edge triggered D flip-flop from NOR gates.
24	Know how to use essential and non-essential prime implicants to find a minimum cost cover for a product-of-sums (POS) implementation.	66	Describe how an asynchronous and synchronous D flip-flop clear and preset work.
25	Give the canonical SOP for a K-map with don't cares.	67	Explain the differences between types of flip-flops and latches.
26	Given two logical functions, draw a logical circuit with multiple output circuit sharing.	68	Design a shift register with D flip-flops.
27	Explain what fan-in means, why high fan-in is a problem, and the engineering trade-offs of avoiding high fan-in.	69	Design a parallel-access shift register with D flip-flops.
28	Simplify a logical function with factoring.	70	Design an asynchronous up-counter or down-counter.
29	Simplify a logical function with disjoint functional decomposition.	71	Design a synchronous up-counter or down-counter.
30	Simplify a logical function with non-disjoint functional decomposition.	72	Design a modulo-n counter with a synchronous reset.
31	Convert a multilevel circuit to NAND or NOR gates.	73	Create a state diagram for a Moore-type finite state machine (FSM).
32	Trace a multi-level circuit to determine its logical function.	74	Derive the state assigned table for a Moore-type finite state machine (FSM).
33	Convert a binary number to decimal.	75	Implement the digital logic circuit for a Moore-type finite state machine (FSM).
34	Convert a binary number to octal and hexadecimal.	76	Demonstrate how different state assignments can have an effect on the cost of a finite state machine (FSM).
35	Design a full adder (FA).	77	Simplify the logical expressions of a finite state machine (FSM) with one-hot encoding.
36	Design an n-bit ripple-carry adder.	78	Design a Mealy-type finite state machine (FSM).
37	Represent a signed binary number in 2's complement form.	79	Design a Mealy-type finite state machine (FSM) serial adder.
38	Add signed binary numbers with 2's complement arithmetic.	80	Design a Moore-type finite state machine (FSM) serial adder.
39	Subtract signed binary numbers with 2's complement arithmetic.	81	Design a finite state machine (FSM) counter with D flip-flops.
40	Design an adder/subtractor unit.	82	Design a finite state machine (FSM) counter with JK flip-flops.
41	Design an adder/subtractor unit that detects arithmetic overflow.	83	Design a finite state machine (FSM) that counts non-sequential pulses on a line.
42	Calculate the critical-path delay for a multi-level circuit.	84	Analyze the behavior of an existing finite state machine (FSM).

The pre-recorded lectures were prepared by recording the PowerPoint slides for each module with a voice over and electronic annotation by the instructor and available online. A pre-class quiz for each module, designed to test the student's comprehension of the module's learning objective, were available on-line and auto-graded. The quizzes were primarily fill-in the blank and usually required the student to design digital logic off-line and then enter the answers on-line. The quizzes represented 25% of the student's grade and had to be completed prior to the beginning of class. Figure 2 is an example of one of the quizzes.



Question

Referring to the basic SR latch constructed with NOR gates as depicted in Figure 7.5(a) and Slide 4 of Module 60, fill-in the following table:

Time	S	R	Q_a	Q_b
t1	0	1	[Qa1]	[Qb1]
t2	0	0	[Qa2]	[Qb2]
t3	1	0	[Qa3]	[Qb3]
t4	0	0	[Qa4]	[Qb4]
t5	0	0	[Qa5]	[Qb5]

Question

Referring to the basic SR latch constructed with NAND gates as depicted in Slide 6 of Module 60, fill-in the following table:

Time	S	R	Q_a	Q_b
t1	0	1	[Qa1]	[Qb1]
t2	0	0	[Qa2]	[Qb2]
t3	1	0	[Qa3]	[Qb3]
t4	0	0	[Qa4]	[Qb4]
t5	0	0	[Qa5]	[Qb5]

Figure 3. Pre-class quiz example

In-Class Problems: Each module had an in-class problem. The in-class problems were designed to take the average student 10-15 minutes to complete. The students did not see the problems until class started and were required to turn them in at the end of class for a grade, which comprised 25% of their grade. An example of an in-class problem is: *Design an n-bit inverter. The circuit will have n number of inputs, x_n through x_1, and n output bits, f_n through f_1. The circuit will have an additional input, s. $f_k = x_k$(not inverted) when s=0 and $f_k = !x_k$(inverted) when s=1, where $1 \leq k \leq n$. Use only XOR gates in your implementation.*

Description of a typical class, fully flipped classroom:
At the beginning of the semester, the students were given a schedule of the modules that would be covered each class period. Prior to each class period, the students were accountable for

studying the module and completing the quiz over the module. They could study the module by watching pre-recorded lectures, reading the sections in the book identified in the module slides, reviewing the slides, researching the learning objective on the internet, or all of these.

In-class activities: During class, the students worked on in-class problems that were turned in at the end of the class. The 50-minute section did two in-class problems while the 75-minute section did three. The students were encouraged to work together to solve the problems, but each student had to turn in their own in-class problem. Students tended to work in groups of 2-4, although some students preferred to work alone. The instructor and undergraduate teaching assistants walked around the room answering questions. These activities took the place of homework, and there was no homework assigned in the class. In-class problems were collected and graded. Occasionally, the instructor might spend 1-2 minutes giving announcements, but usually the students began working on the problems and asking questions as soon as they were displayed on the projector screens, which was frequently 5-10 minutes before class started.

Exams: Four open-book, open-note non-comprehensive exams were given. The students were allowed to use their phones, tablets, or laptops to access their notes on the in-class problems and pre-class quizzes, lecture slides, and lecture videos. The first three exams were given during the 120-minute discussion periods and covered Modules 1-20, 21-38, and 39-56, respectively. The last exam was given during the class's 150-minute final exam period and covered Modules 57-84. The exams were equally weighted and together represented 25% of the student's grade. An exam review was also conducted by an undergraduate teaching assistant a week before each exam was given. In order to track student learning, similar (but not the same) questions to those in the first semester, pre-flipped class, were asked.

Evaluation

Learning Objective Achievement

Performance on exam questions linked to the same learning objective were examined for three of the four exams- future work will include the analysis of the fourth and final exam. The Independent Samples Mann-Whitney U-test was used to test for significant differences between performance across semesters. SPSS was used to conduct all statistical analysis.

COPUS Observations

The Classroom Observation Protocol for Undergraduate STEM (COPUS) (Smith, Jones et al. 2013) was used to quantify how professors and students were spending time during class. To perform the COPUS, a trained observer visited each course three times in a two-week period. For each COPUS observation, the observers indicated in each two-minute interval of class time whether or not 13 student and 12 instructor behaviors (listed in Table 2) occurred. Data from the three observations of each course were averaged, and then the thirteen behaviors were collapsed into four categories. For students, those categories were *receiving*, *working* (included individual thinking/working, clicker question in groups, working in groups, and other group), *talking to class* (answering questions, asking questions, whole class discussion, and student presentation), and *other* (waiting, other). For instructors, those categories were *presenting* (lecturing, real-time

writing, and demo/video), *guiding* (follow-up, posing question, clicker question, answering question, moving around to groups, and one-on-one), and *other* (waiting, other).

Table 1. Student and instructor behaviors tracked on COPUS observation

COPUS Student Behaviors	COPUS Instructor Behaviors
Listening [L]	Lecturing [Lec]
Answering Questions [AnQ]	Real-time writing [RtW]
Asking Questions [SQ]	Demo/Video [D/V]
Whole class discussion [WC]	Follow-up [Fup]
Student Presentation [SP]	Posing question [PQ]
Individual thinking/working [Ind]	Clicker question [CQ]
Clicker question in groups [CG]	Answering Question [AnQ]
Working in groups [WG]	Moving around to groups [MG]
Other group [OG]	One-on-One [1o1]
Predicting the outcome of something [Prd]	Administrative [Adm]
Test/Quiz [TQ]	Waiting [W]
Waiting [W]	Other [O]
Other [O]	

Student Surveys

A mid-semester survey was given in order to understand student opinions about the operation of the flipped classroom model. The survey was administered online using Qualtrics. The survey was not anonymous, because students were given extra credit for completing it. However students were informed that the results of the survey would only be seen by the post-doctoral teaching fellow assisting with this project, and the instructor of the course would only see the aggregated results. Surveys asked Likert-scale questions about the effectiveness/usefulness of each course component, and how they prepare for class.

Results and Discussion

Learning Objective Achievement

Exam results showed a significant improvement in performance on several learning objectives, particularly on exams 1 and 3. On exam 1, there was a statistically significant improvement on 10/14 learning objectives. On exam 2, there was a statistically significant improvement on 3/13 learning objectives, but a statistically significant decrease in score on 4/13 learning objectives. On exam 3, there was a statistically significant improvement on 9/13 learning objectives (Figure 4). Across the three exams, there was an improvement on 21/40 learning objectives (53%), and a decrease on 4/40 learning objectives (10%).

Exam 2 revealed declines in learning objectives 24, 25, 29, 30, 33, and 37 (see Figure 4).

Figure 4. Learning Objective Achievement Across Semesters. Fall 2017 (blue)= hybrid active learning and lecture; Spring 2018 (orange)= fully flipped classroom.

Learning objective 24, *"Know how to use essential and non-essential prime implicants to find a minimum cost cover for a product-of-sums (POS) implementation,"* covered POS implementations in one module whereas its dual concept, SOP (sum-of-products), was covered in five modules. Possibly a more detailed explanation of POS in the video lecture and slides could improve the student's understanding of this concept.

The learning objective for Module 25 was "*Give the canonical SOP for a K-map with don't cares"*. This is a key concept used in later modules. Thus, expansion of this concept in the video lecture and slides maybe warranted. Learning objectives 29 and 30 cover the rather arcane and difficult concept of functional decomposition. Understanding functional decomposition is probably more important than understanding the rather esoteric difference between disjoint and non-disjoint functional decomposition. These two modules should probably be revised to focus more on how to perform functional decomposition than the difference between disjoint and non-disjoint functional decomposition. The ability to convert a binary number to decimal (Learning Objective 29) is a fundamental concept in digital logic design. It is introduced in middle and high school, so assumptions were made that this would be a review topic. A more thorough review of this concept is probably needed.

Module 37 is the beginning of a series of modules on 2's complement arithmetic. Since the students were just beginning to understand this idea, it may have been too soon to test their knowledge on an examination. In the previous offering of the course (Fall 2017), this concept was not tested until several weeks after being introduced to the concept, where in Spring 2018, it was tested only a few days after being introduced to the concept. Moving it to the next examination might give a better view of whether they were grasping this concept or not.

A factor that might have contributed to the apparent improvement in comprehension between the two semesters is the amount of time the students had to work on exam questions. In the Fall semester, the students had 300 minutes to answer 54 questions (about 5.6 minutes per question) and in the Spring they had 510 minutes to answer 64 questions (about 8 minutes per question). On the other hand, most of the Fall students were unable to complete one of the three exams in the 75 minute period, while the completion rate of the four Spring exams was very consistent with all of the students completing the exams within the allotted time. Thus, it is more likely that 5.6 minutes per question was not enough time to answer the questions properly whereas 8 minutes was.

Another factor that could have contributed to the improvement in scores is if students used old exams to study. In the Fall, students were given the exam solutions for 32 of the 64 questions re-used in the Spring exams. Of course, the 32 questions were re-worded and different values were used, but this could have also contributed to the apparent growth in comprehension. However, the practice of giving past exams for students to study is used in other courses and probably wasn't a contributing factor. The decline in several objectives in Exam 2 is an indication that the students did not study old exams, or if they did, they were not helpful.

COPUS Observations

COPUS observations indicate that the class indeed was fully flipped, with students doing 97% working and 3% listening in Spring 2018, compared to 45% "working" and 45% "listening" in Fall 2017. The instructor spent nearly all (99%) of class time "guiding," compared to 62% in Fall 2017 (Figure 5). The COPUS results show that the flipped-class did indeed not include any lecture. The instructor purposely did not lecture at all. This leads to the question, *were the students really prepared to work without any lecture*? Subjectively, the instructor felt that they were. The fact that the students began working on the problems and asking questions about them as soon as they were displayed on the monitor is an indication that they were ready. Also, the students worked on the problems in informal teams. Many times one of the team members would explain the concept to the others. In general, students who had difficulties on one concept were not the same ones who had difficulties on other concepts.

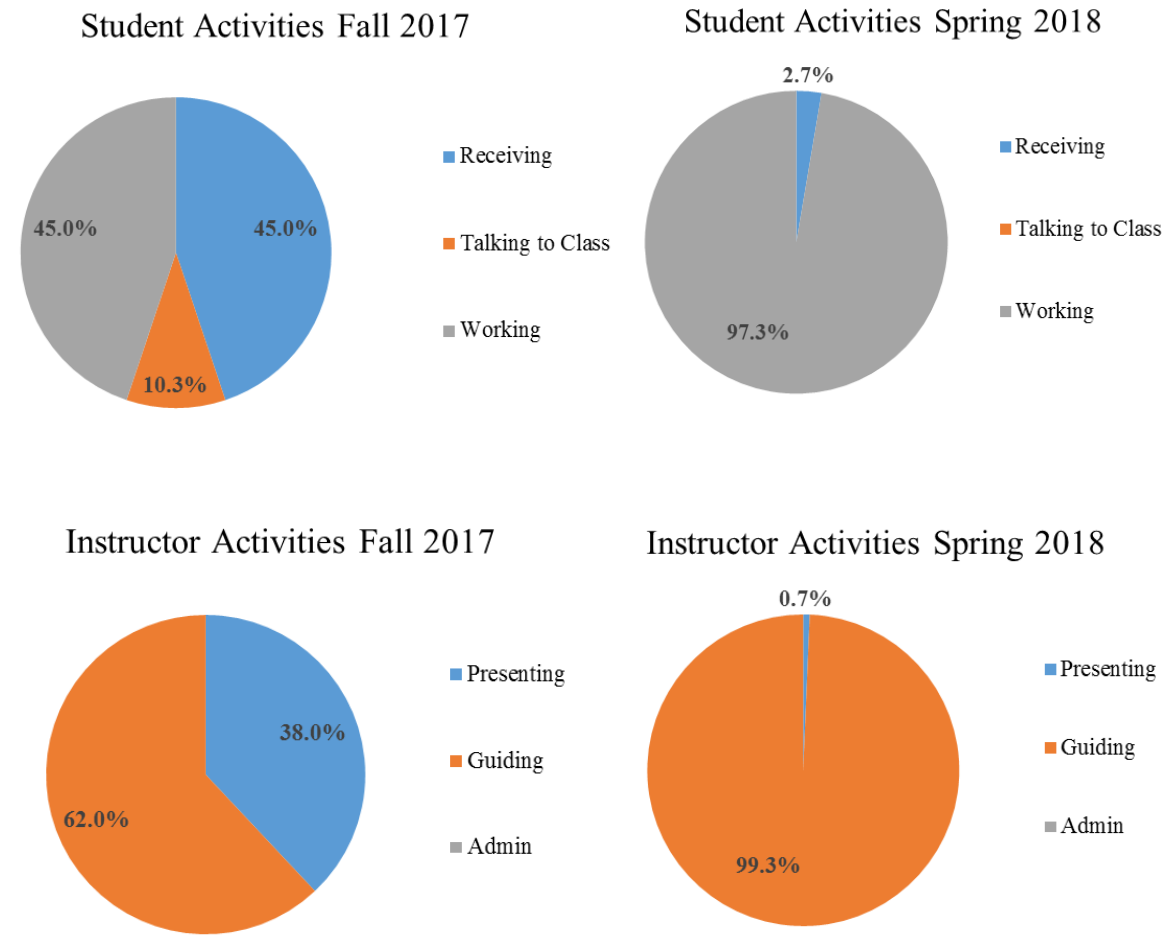


Figure 5. COPUS results Fall 2017 (hybrid active and lecture) compared to Spring 2018 (fully flipped).

Student Surveys

Students indicated that they came to class prepared (over 90% of respondents), and the most common way to prepare for class was by "reviewing the PowerPoint slides," (43%), and "watching videos," (34%) (Figure 6). Students generally agreed that each component of the course was effective, but the most effective component was the in-class problems (90% of respondents "agreed" they were useful). Figure 7 shows these results.

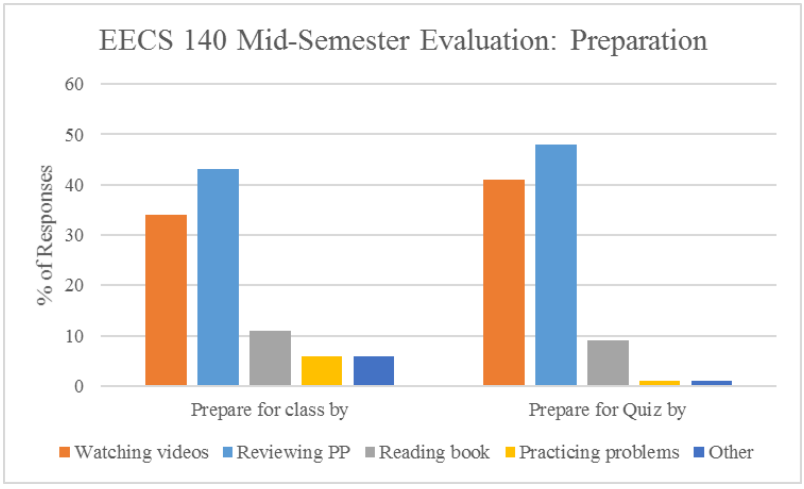


Figure 6. Self-reported preparation methods for flipped classroom.

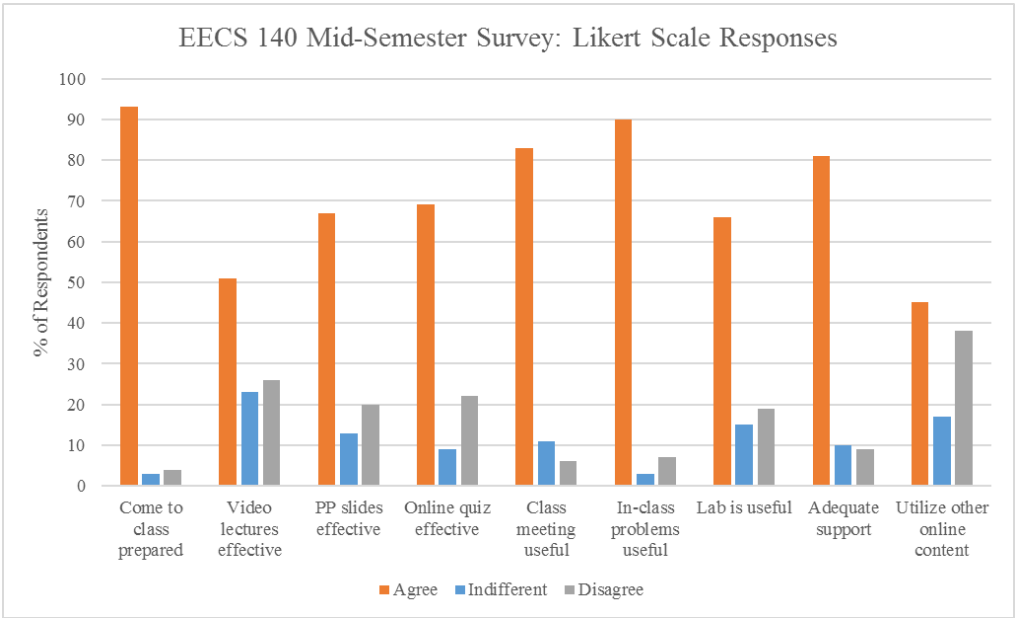


Figure 7. Mid-semester survey results for flipped classroom.

Conclusions

The fully-flipped implementation of this course resulted in significantly better learning outcomes for students. The instructor reported being more in tune with the students and their understanding of concepts, and is motivated to attend to the modules where students are still struggling. While much effort went into creating the online lecture materials to allow for so much active learning, it resulted in more engaged students, higher achievement of learning outcomes, and a more highly satisfied instructional team. Future work includes the analysis of learning outcomes on exam 4, and the development of new modules to more clearly address concepts that were difficult for students.

Acknowledgements

This material is based upon work supported by th National Science Foundation under Grant Number DUE1525775. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Felder, R. M., D. R. Woods, J. E. Stice and A. Rugarcia (2000). "The future of engineering education II. Teaching methods that work." Chemical Engineering Education **34**(1): 26-39.

Freeman, S., S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt and M. P. Wenderoth (2014). "Active learning increases student performance in science, engineering, and mathematics." Proceedings of the National Academy of Sciences **111**(23): 8410-8415.

Smith, M. K., F. H. Jones, S. L. Gilbert and C. E. Wieman (2013). "The Classroom Observation Protocol for Undergraduate STEM (COPUS): a new instrument to characterize university STEM classroom practices." CBE-Life Sciences Education **12**(4): 618-627.