Using Practical Examples in Teaching Digital Logic Design

Dr. Joseph P Hoffbeck, University of Portland

Joseph P. Hoffbeck is an Associate Professor of Electrical Engineering at the University of Portland in Portland, Oregon. He has a Ph.D. from Purdue University, West Lafayette, Indiana. He previously worked with digital cell phone systems at Lucent Technologies (formerly AT&T Bell Labs) in Whippany, New Jersey. His technical interests include communication systems, digital signal processing, and remote sensing.

Using Practical Examples in Teaching Digital Logic Design

Abstract

Digital logic design is often taught from the bottom up starting with the simplest components (transistors and gates), proceeding through combinational and sequential logic circuits, and if there is time may finish up with the basic components of microprocessors. With the bottom up approach, it may be a fairly long time before students see a complete system that performs a recognizable function. Most of the standard example circuits, such as binary adders, decoders, multiplexers, etc., are parts used in a larger system. While knowledge of the standard circuits is crucial for building more complex circuits, these standard circuits might not capture the students' interest as much as a complete system. Therefore, this paper describes three proposed example circuits that are simple enough to cover in the first logic design course, but yet are complete systems that perform useful functions. The proposed circuits are a game show buzz-in system that determines which of two contestants rings in first, a standard 12-hour digital clock, and a car alarm that could honk a car horn if someone enters the car without resetting the alarm. The proposed circuits can be used as examples or homework problems in addition to the standard circuits to increase students' interest in the material and to show how useful the design techniques can be. Surveys were given to the students after the proposed circuits and the standard circuits were covered to assess the level of student interest generated by the examples. The results of the surveys are presented in the paper along with detailed descriptions of the circuits.

Background

There have been many previous papers describing methods for teaching logic design using breadboards[1], using software to aid in the design of digital circuits[2-5], using FPGA or CPLD to implement designs[6-14], and using remote implementation of digital circuits[15]. Regardless of how the circuits are implemented, it is desirable that the circuits be interesting to the students.

This paper presents three of the example circuits that are used in a three credit hour lecture course called Digital Logic Design. This introductory, semester-long course consists of standard lectures along with in-class exercises where the students work in groups to analyze and design digital circuits. There is no lab component of the course, so one of the goals of the examples is to tie the material to real world systems as much as possible. The course has no prerequisites and is taken by sophomore electrical engineering and computer science majors.

Unlike most traditional example circuits, each of the presented circuits is a complete system that solves a real world problem. The circuits are designed to show how the material can be used in the real world, yet are simple enough to be included in the first logic design course. The effectiveness of the circuits was measured using student surveys and was compared to the results of more traditional example circuits.

Game Show Circuit

Many game shows use a circuit to determine which of the contestants ring in first. This circuit makes a good example for a sequential logic circuit because it performs a recognizable function with which students are familiar, yet it is quite simple. A circuit can be designed to determine which of two contestants rings in first. It has two inputs S_1 and S_0 which are connected to the contestants' buttons. The circuit has two outputs Z_1 and Z_0 which are connected to LED's to indicate which contestant rang in first. There is also a reset button that is used by the game show host to asynchronously reset the flip-flops to the initial state before each question. If contestant 0 rings in first, the circuit turns on LED 0. Once LED 0 is on, the circuit leaves it on regardless of the inputs until the circuit is asynchronously reset by the game show host. If contestant 1 rings in first, the circuit turns on LED 1 and leaves it on until the circuit is reset. If there is a tie, both LED's are turned on.

The circuit requires four states: reset, contestant 0 wins, contestant 1 wins, and tie. One way to map the states is to use state 00 for reset, state 01 for contestant 0 wins, state 10 for contestant 1 wins, and state 11 for a tie. With this mapping, the outputs are equal to the current state, which simplifies the output equations.

In the state table shown in Table 1, q_1q_0 is the current state, $q_1{*}q_0{*}$ is the next state, and Z_1Z_0 is the current output. There are four columns of values for the next state $q_1{*}q_0{*}$. The first column contains the values of the next state when $S_1S_0 = 00$, the second column contains the next state when $S_1S_0 = 01$, and so on.

Table 1: State Table for Game Show Circuit

q_1q_0	$q_1{*}q_0{*}$ $S_1S_0 =$ 00 01 10 11	Z_1Z_0
00	00 01 10 11	00
01	01 01 01 01	01
10	10 10 10 10	10
11	11 11 11 11	11

It can be seen from Table 1, after the circuit is reset to state 00, it will stay in state 00 and both LED's will be off as long as neither contestant presses a button (that is as long as $S_1S_0 = 00$). If contestant 0 is the first to press the button ($S_1S_0 = 01$), the circuit goes to state 01, and the circuit stays in state 01 regardless of the inputs until the game show host resets the circuit using the asynchronous reset. While the circuit is in state 01, the output Z_0 is 1 and the output Z_1 is 0, and so LED 0 is on and LED 1 is off to show that contestant 0 rang in first. Likewise if contestant 1 presses the button first ($S_1S_0 = 10$), the circuit goes to state 10 and turns LED 1 on and turns LED 0 off, and the circuit stays in state 10 regardless of the inputs until an asynchronous reset. If both inputs are pressed at the same time, the circuit goes to state 11 and both LEDs are turned on to indicate a tie.

Using K-maps, the excitation and output equations for D flip-flops can be found to be as follows:

$D_0 = q_0 + S_0 q_1'$
$D_1 = q_1 + S_1 q_0'$
$Z_0 = q_0$
$Z_1 = q_1$

In the circuit diagram shown below in Figure 1, Vcc is 9 Volts, and a 555 timer is used to generate the clock signal with a frequency of about 100 kHz. The values of R2, R3, and C can be changed to generate other clock frequencies (see the 555 datasheet for details). It is unlikely that a tie will occur if the clock frequency is very high.

A breadboard with the circuit was passed around during the lecture so that students could see the circuit in operation (see Figure 2).

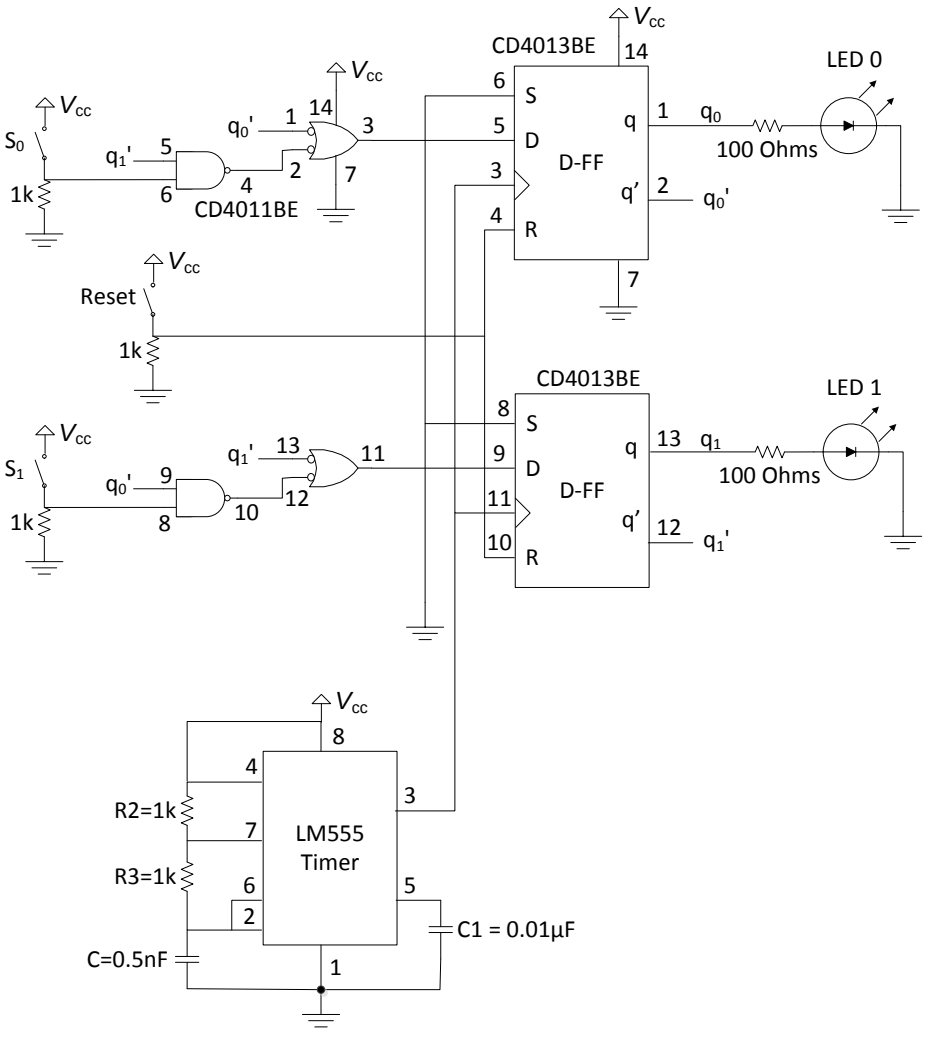

Figure 1: Schematic of Game Show Circuit

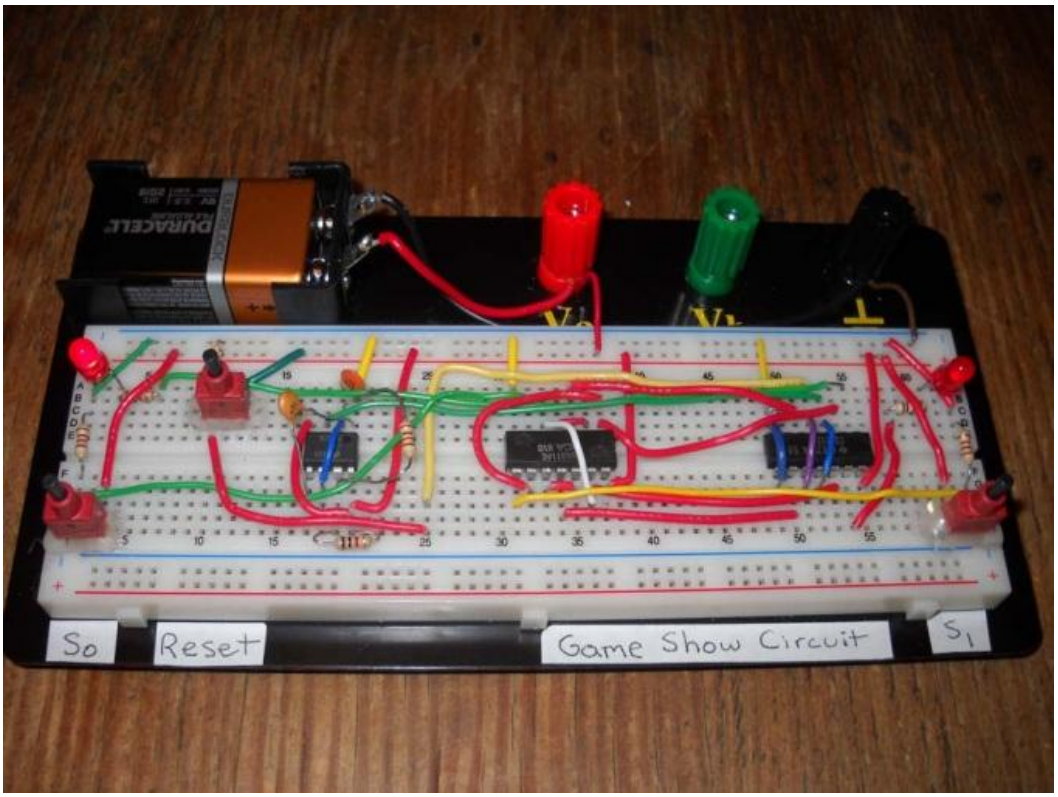

Figure 2: Breadboard with Game Show Circuit

12-Hour Clock

The 12-hour clock is a standard commercial product. It performs a useful function, yet it is simple enough that it can be used as either an example in lecture, or can be designed by the students as a series of homework problems or in project. In the author's course, part of the clock is presented as an example in lecture, and the students design the rest of the clock in a group project and homework problems.

Digital clocks are usually set up to start at 12:00, and they count 12:01, 12:02, 12:03, 12:04, 12:05, 12:06, 12:07, 12:08, 12:09, 12:10, and eventually the clock gets to 12:58, 12:59, 1:00, and so on. The one's place of the minutes (the right-most digit) counts 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and then repeats, and a circuit that counts in this way is called a mod-10 counter. The ten's place of the minutes (second digit from the right) counts 0, 1, 2, 3, 4, 5, and then repeats, which is called a mod-6 counter. The hour counter counts 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and repeats. One way to design the clock is to break it up into smaller parts as shown in Figure 3 below.

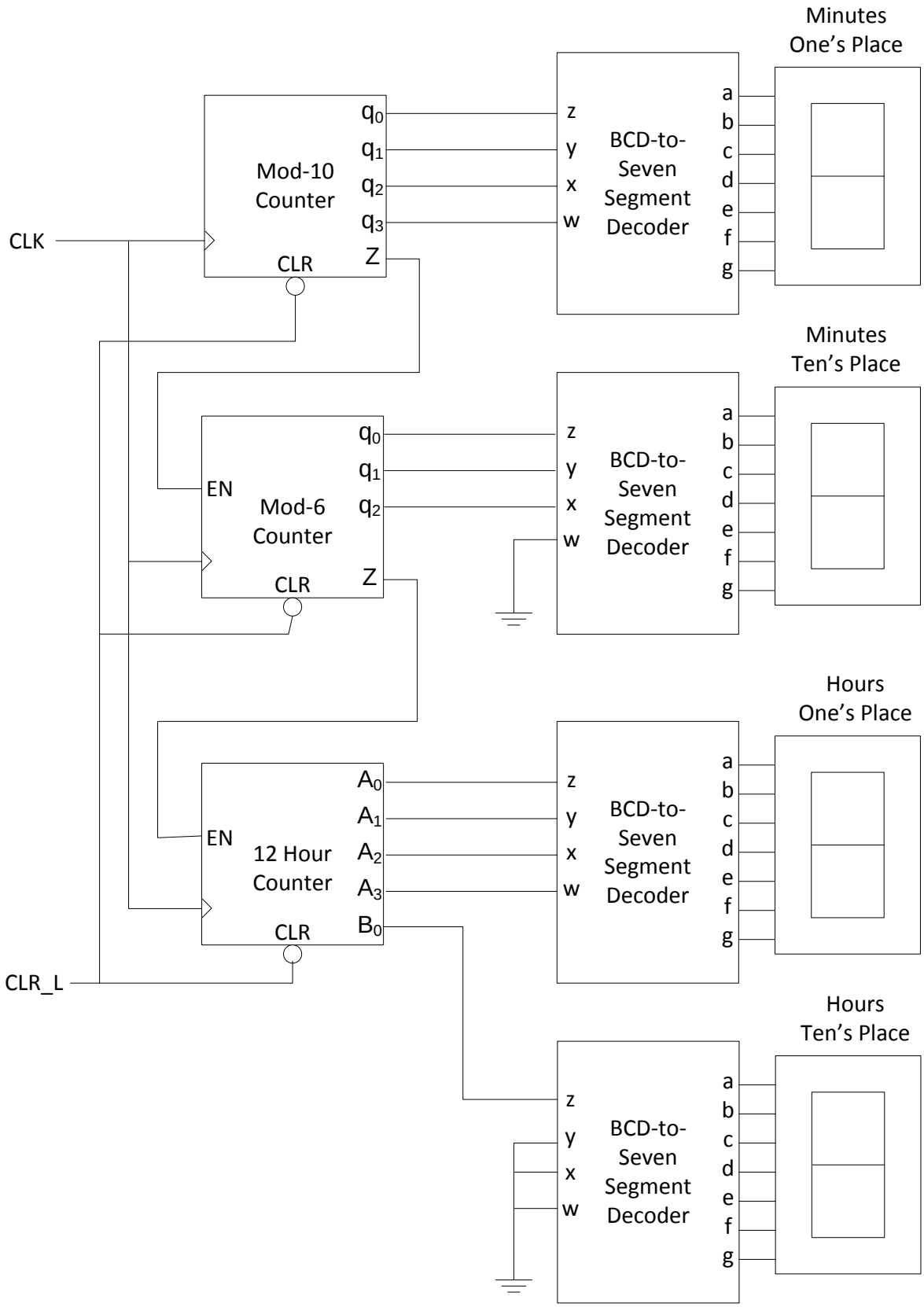

Figure 3: Block Diagram of 12-Hour Clock

The output from each counter is a binary coded decimal (BCD) number that represents one of the digits in the time, and BCD-to-Seven segment decoders are used to drive the seven segment displays.

This circuit is a clocked synchronous circuit where all of the flip-flops have the same clock signal. If the clock signal is set up to have one pulse per minute, the mod-10 counter will increment every minute and needs to have an output Z that indicates when it is at the maximum count (1001). Since the mod-6 counter only increments every 10 minutes, it needs to have an enable input which is connected to the maximum count indicator from the mod-10 counter. Furthermore, the mod-6 counter needs to have a maximum count indicator Z that is 1 when the mod-6 counter is at the maximum count (0101) and it is enabled. The hour counter needs to have an enable input which is connected to the maximum count indicator from the mod-6 counter.

The state table for the mod-10 counter is shown in Table 2. Don't care conditions (x's) were placed in the unused states to help simplify the excitation equations.

Table 2: State Table for Mod-10 Counter

$q_3q_2q_1q_0$	$q_3^*q_2^*q_1^*q_0^*$	Z
0000	0001	0
0001	0010	0
0010	0011	0
0011	0100	0
0100	0101	0
0101	0110	0
0110	0111	0
0111	1000	0
1000	1001	0
1001	0000	1
1010	xxxx	x
1011	xxxx	x
1100	xxxx	x
1101	xxxx	x
1110	xxxx	x
1111	xxxx	x

The excitation equations for D flip-flops and the output equation for the mod-10 counter were found using K-maps and are shown below.

$D_0 = q_0'$
$D_1 = q_3'q_1'q_0 + q_1q_0'$
$D_2 = q_2'q_1q_0 + q_2q_0' + q_2q_1'$
$D_3 = q_3q_0' + q_2q_1q_0$
$Z = q_3q_0$

The state table for the mod-6 counter is shown in Table 3, and the excitation and output equations for the mod-6 counter are shown below.

$D_0 = q_0'E + q_0E' = q_0 \oplus E$
$D_1 = q_2'q_1'q_0E + q_1E' + q_1q_0'$
$D_2 = q_2E' + q_2q_0' + q_1q_0E$
$Z = q_2q_0E$

Table 3: State Table for Mod-6 Counter

$q_2q_1q_0$	$q_2^*q_1^*q_0^*$		Z	
	E=0	E=1	E=0	E=1
000	000	001	0	0
001	001	010	0	0
010	010	011	0	0
011	011	100	0	0
100	100	101	0	0
101	101	000	0	1
110	xxx	xxx	x	x
111	xxx	xxx	x	x

The hour counter has 12 states, and so it requires four flip-flops. Since the hour counter has four flip-flops and one input, the excitation equations are functions of five variables. In order to avoid requiring five-variable K-maps, the enable function was implemented separately by designing a flip-flop with an enable input as shown in Figure 4. When EN = 0, the multiplexer selects the current state q and applies it to the input of the flip-flop so that the next state will be the same as the current state ($q^* = q$). When EN = 1, the multiplexer selects the input D and applies it to the input of the flip-flop so that the circuit behaves like a standard flip-flop.

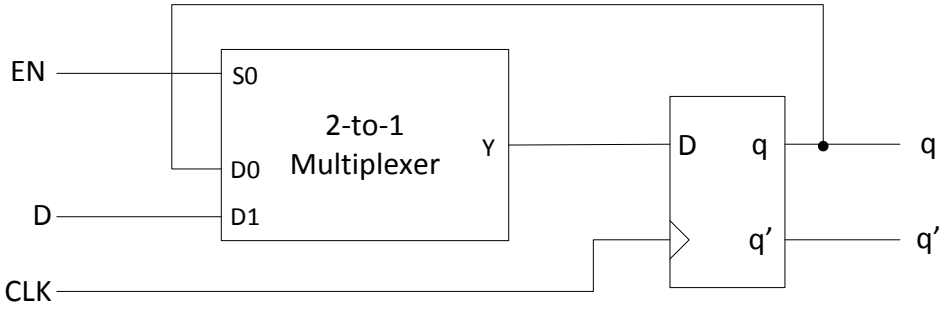

Figure 4: D Flip-Flop with Enable

Therefore the hour counter was designed without an enable input so that the excitation equations are functions of only four variables, but it was constructed using flip-flops with enable. As shown in Table 4, the state counts from 0000 to 1011, and the output functions generate BCD numbers for the one's place ($B_3B_2B_1B_0$) and for the ten's place (A_0). Note that only one bit is needed for the ten's place because its value is either 0 or 1.

Table 4: State Table for Hour Counter (No Enable)

$q_3q_2q_1q_0$	$q_3^*q_2^*q_1^*q_0^*$	A_0	$B_3B_2B_1B_0$	Display
0000	0001	1	0010	12
0001	0010	0	0001	01
0010	0011	0	0010	02
0011	0100	0	0011	03
0100	0101	0	0100	04
0101	0110	0	0101	05
0110	0111	0	0110	06
0111	1000	0	0111	07
1000	1001	0	1000	08
1001	1010	0	1001	09
1010	1011	1	0000	10
1011	0000	1	0001	11
1100	xxxx	x	xxxx	
1101	xxxx	x	xxxx	
1110	xxxx	x	xxxx	
1111	xxxx	x	xxxx	

The excitation and output equations are shown below.

$D_0 = q_0'$
$D_1 = q_1 \oplus q_0$
$D_2 = q_3'q_2'q_1q_0 + q_2q_0' + q_2q_1'$
$D_3 = q_2q_1q_0 + q_3q_1' + q_3q_0'$
$A_0 = q_3'q_2'q_1'q_0' + q_3q_1$
$B_0 = q_0$
$B_1 = q_3'q_2'q_0' + q_3'q_1$
$B_2 = q_2$
$B_3 = q_3\, q_1'$

The clock signal, which needs to have one pulse every minute, can be generated by dividing the signal from a crystal oscillator down to the required frequency. The time can be set by having a button that allows the user to temporarily select a clock that has a higher frequency so that the clock counts quicker than normal.

In order to demonstrate the circuit, the 12-hour clock was implemented using a custom integrated circuit fabricated through the MOSIS program[16] and four BCD-to-seven segment decoders. A breadboard with the circuit was circulated around the class (Figure 5).

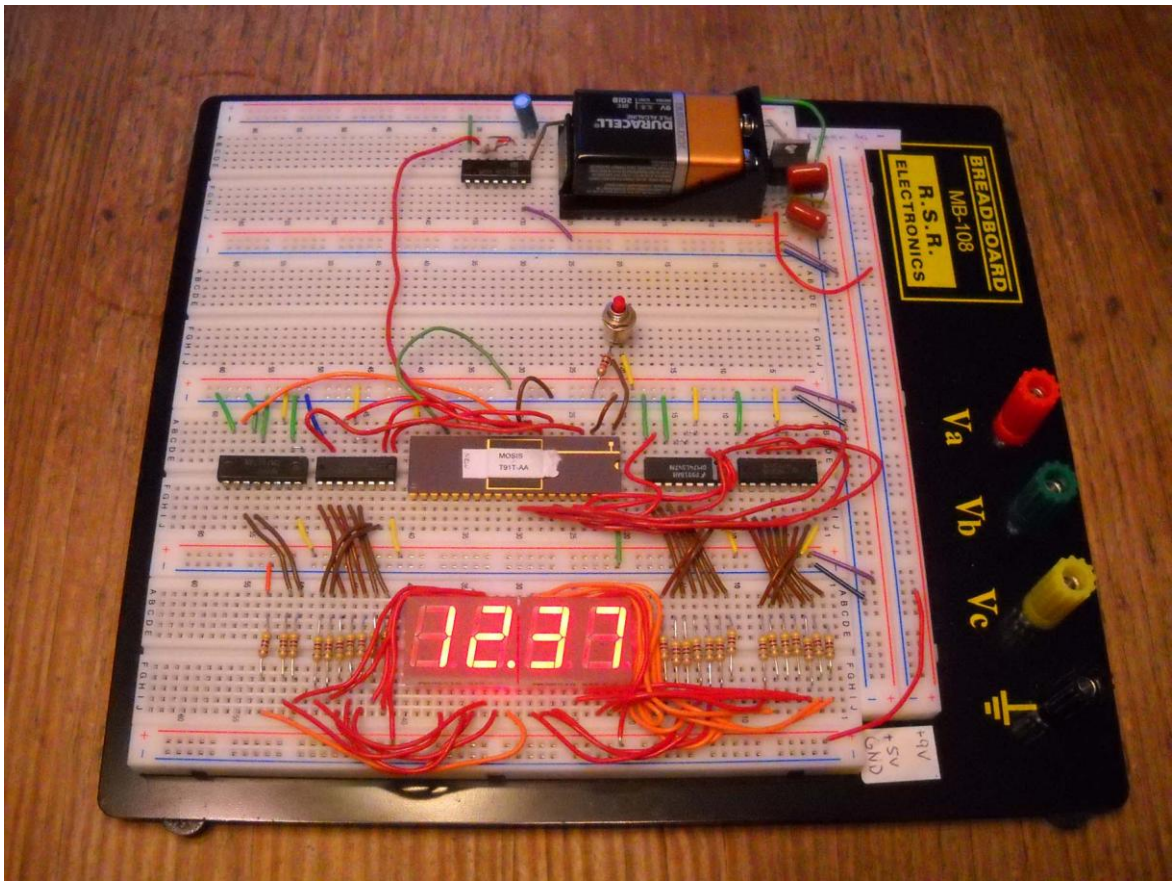

Figure 5: Breadboard with 12-Hour Clock

Car Alarm

A fairly simple circuit can be designed that could operate a car alarm. The circuit has one input Y which would be connected to the car's door switch to determine if the car door is open or shut. When the door is shut Y = 0, and when the door is open Y = 1. The circuit has one output Z which is used to operate a relay that honks the horn by shorting the wires that go to the horn switch in the steering wheel. When Z = 1, the relay is activated and the horn honks. The circuit would be asynchronously reset by the accessories power line that is high when the ignition is turned on or is in accessory-only mode, both of which require the key to the car.

The state table is shown in Table 5 below. While the ignition is on, the flip-flops are forced into state 000 by the asynchronous reset. Then, when the ignition is turned off, the circuit stays in state 000 while door is closed (Y = 0) to wait for the driver to get out of the car. When the driver opens the door (Y = 1), the circuit goes to state 001. The circuit stays in state 001 while the door is open (Y = 1), and when the driver closes the door (Y = 0) the circuit proceeds to state 010. At this point, the alarm is "armed" in the sense that it is waiting for the door to open and to honk the horn. As long as the door is closed (Y=0), the circuit stays in state 010, but when the door is opened (Y = 1), the circuit moves to state 011. Once in state 011, the circuit does not honk the horn yet because usually it is the driver returning to the car. If the clock is set to have

one pulse for every 20 seconds, there will be at least a 20 second delay before the circuit moves to state 100 and honks the horn. The driver would use this time to insert the key into the ignition and to turn the ignition on, which asynchronously resets the circuit and prevents the horn from honking. If the ignition is not turned on within 20 seconds, the circuit proceeds to state 100 regardless of whether the door is open or not, and the horn is honked (Z = 1). The circuit will stay in state 100 and honk the horn until it is reset by the ignition or the battery dies.

Don't cares (x's) were placed in the unused states to help simplify the excitation and output equations.

Table 5: State Table for Car Alarm Circuit

$q_2q_1q_0$	$q_2^*q_1^*q_0^*$ Y = 0	Y = 1	Z
000	000	001	0
001	010	001	0
010	010	011	0
011	100	100	0
100	100	100	1
101	xxx	xxx	x
110	xxx	xxx	x
111	xxx	xxx	x

The excitation and output equations were found using K-maps as follows.

$D_0 = q_1'q_0Y + q_2'q_0'Y$
$D_1 = q_1'q_0Y' + q_1q_0'$
$D_2 = q_1q_0 + q_2$
$Z = q_2$

In order to be practical, this circuit would also need to latch the input so that the circuit will notice if someone opens the door, jumps inside, and closes the door in less than 20 seconds. (It would also be desirable to redesign the circuit so that it turns the horn on and off repeatedly instead of just turning the horn on constantly, and to automatically reset the circuit after the horn is honked for say five minutes, but these features require many more states and a more complicated circuit.)

Assessment

In an attempt to measure how interesting and effective the students find these examples, a survey was given after each example was completed. The survey asked "Did you find the example circuit interesting?" and "Was the example circuit helpful?". The students scored each question on a five point scale where 1 was not at all interesting/helpful, and 5 was very interesting/helpful. The survey also asked for optional comments about each example circuit. The survey was also given after two conventional examples: 4-bit binary adder and 3-to-8 binary decoder. For each question, the average student rating and the number of responses (N) are shown below in Table 6.

Table 6: Average Scores from Student Survey

Example	N	Interesting	Helpful
Binary Adder	33	4.2	4.5
Binary Decoder	31	3.9	4.1
Game Show Circuit	27	4.3	4.2
Digital Clock	29	4.5	4.4
Car Alarm	30	4.4	4.1

Comments from each example circuit are summarized below.

Binary Adder:

- More like this!
- The example was made much more interesting when you gave examples of devices it's used in. (What actually makes this class awesome to me is solving practical problems with solutions that are used (or seem to be)).
- It was very interesting to see how these things work in an easily laid out format.
- I find this stuff very interesting

Binary Decoder:

- I'm still lost on some parts of the decoder
- Hierarchical design is difficult to grasp with decoders
- It was cool
- Understanding how to put the circuits together for decoders, etc. is more challenging than the beginning course content; takes more time to grasp
- Interested in alternative ways circuit could be made. Also if there are benefits to the different methods.

Game Show Circuit:

- Cool real-life application
- I always wondered how it worked.
- Not much function, but a good example problem
- Fun to problem to solve. Interesting thought and…cool!

Digital Clock:

- Cool example
- It was very cool
- Good!
- Some of the underlying work would be nice to see
- It would have been helpful to see some k-maps done

Car Alarm:

- Entertaining, practical example
- It was really cool to see a relevant example used in real life
- Very cool
- I thought this was my favorite example.
- I didn't realize you could do this with sequential logic. Good example.

The students rated the game show circuit, digital clock, and car alarm as slightly more interesting than the conventional examples binary adder and binary decoder. However, those circuits may not be directly comparable to the binary adder and binary decoder because the game show circuit, digital clock, and car alarm are all sequential logic circuits whereas the binary adder and binary decoder are combinational logic circuits.

The binary adder and the digital clock were rated as being slightly more helpful than the others. This result may have occurred because the students designed a binary adder and parts of the digital clock as homework assignments and in a group project, so these two examples were directly related to problems that the students were required to solve. However, the students also designed a large binary decoder in the homework, but that example was not rated as high in helpfulness. This result might be explained by the fact that at the time of the survey, the students were struggling with understanding the decoder, as is made clear in the comments.

Overall, the students rated the game show circuit, digital clock, and car alarm as being highly interesting and helpful. These circuits were rated as slightly more interesting than the more conventional circuits, but they were not necessarily rated as being more helpful to them.

Conclusion

Three example circuits were described for a digital logic design course. Although the examples were relatively simple, they were complete systems that solve real-world problems. A survey found that the students seemed to find the complete system examples slightly more interesting than the more conventional circuits, but not necessarily more helpful.

References

1. Show them NAND gates and they will come, Barrett, S.F.; Hamann, J.; Coon, D.; Crips, P.M.; Pierre, J., Computers in Education Journal, v 17, n 2, p 26-36, April/June 2007.

2. A Builder and Simulator Program with Interactive Virtual Environments for the Discovery and Design of Logic Digital Circuits, Miguel-de-Priego, Arturo, 2013 Frontiers in Education Conference (FIE2013), Oklahoma City, Oklahoma, October 23-26, 2013.

3. Visiboole: Transforming digital logic education, Devore, John J.; Soldan, David L., ASEE Annual Conference and Exposition, ASEE2012, June 10, 2012 - June 13, 2012.

4. Teaching digital logic design using the GOAL (Guided On-demand Adaptive Learning) system, Williams, Ronald D.; Dugan, Joanne Bechta, ASEE Annual Conference and Exposition, June 26, 2011 - June 29, 2011.

5. Toward an interactive environment for embedded systems design, Obeidat, Fadi; Alkhasawneh, Ruba; Tucker, Jerry; Klenke, Robert, ASEE Annual Conference and Exposition, June 20, 2010 - June 23, 2010.

6. The CPLD Provides a Third Option in the Introductory Circuits Course, Hill, Jonathan; Yu, Ying, ASEE Annual Conference and Exposition, Conference Proceedings, June 10 - 13, 2012.

7. Developing Undergraduate FPGA Curriculum using Altium Software and Hardware, Mayer, Erik A., Computers in Education Journal, v 23, n 1, p 35-42, January-March 2013.

8. FPGArcade: Motivating the study of digital hardware, Neebel, Danial J.; Burek, Nicholas J.; Griebel, Thomas, ASEE Annual Conference and Exposition, June 10 - 13, 2012.

9. ARM/FPGA/I2C sensor network development and teaching platform, Mondragon, Antonio Francisco; Purohit, Prafull, ASEE Annual Conference and Exposition, June 26 - 29, 2011.

10. Interdisciplinary laboratory projects integrating LabVIEW with VHDL models implemented in FPGA hardware, Hayne, Ronald; McKinney, Mark, ASEE Annual Conference and Exposition, June 20 - 23, 2010.

11. A LabVIEW FPGA toolkit to teach digital logic design, Perales, Troy; Morgan, Joseph; Porter, Jay, ASEE Annual Conference and Exposition, June 14 - 17, 2009.

12. Collaborative project-based learning to enhance freshman design experience in digital engineering, Dong, Jianyu; Warter-Perez, Nancy, ASEE Annual Conference and Exposition, June 14 - 17, 2009.

13. Extensive use of advanced FPGA technology in digital design education, Radu, Mihaela; Cole, Clint; Dabacan, Mircea Alexandru; Sexton, Shannon, ASEE Annual Conference and Exposition, June 22 - 24, 2008.

14. Introducing field-programmable gate arrays into sophomore digital circuits course, Sin, Ming Loo; Planting, Arlen; Murdock, Matt, ASEE Annual Conference and Exposition, June 18 - 21, 2006.

15. Boole-WebLab-Deusto: Integration of a Remote Lab in a Tool for Digital Circuits Design, Garcia-Zubia, Javier; Rodriguez-Gil, Luis; Orduña, Pablo; Angulo, Ignacio; Dziabenko, Olga, 2013 Frontiers in Education Conference, October 23 - 26, 2013.

16. The MOSIS Service, www.mosis.com, accessed December 22, 2013.