On Line Assignment and Laboratory System for Digital Logic Timing Diagrams

***Daniel Kohn*[1]**

Abstract – With more universities turning toward online courses, educators are turning to open source scripts to create interactive, random assignments and labs for their students. When it comes to digital logic courses that use flip-flops and latches, the automatic generation of interactive assignments becomes a problem. How do you generate random timing diagrams for the inputs to these devices? How do you make it so a student submits the resulting timing diagram and have it accepted by an HTML form? Furthermore, how do you check the student's answer via a script so you can provide instant feedback? This paper describes one such effort to create an assignment/lab script that will allow an instructor to create interactive assignments and/or laboratories appropriate for these topics.

Keywords: Digital Logic, On Line Assessments, Scripts, Digital Timing Diagrams, PHP.

INTRODUCTION

With many universities turning toward online courses to increase enrollment and revenue, online courses are becoming more and more prevalent. Issues have arisen concerning on-line testing and assignments, since supervision of students while completing assignments and tests is difficult if not impossible. Cheating has become a major concern for courses delivered in an on-line environment. Many instructors have chosen to randomize questions to insure the students do their own work. But when it comes to digital logic timing diagrams this has been very difficult. Most online assignment and testing software forces instructors to create test banks and then give students random questions from the pre-defined bank. The bank of questions is time consuming to generate and also limited on the type of questions that can be presented. Many online test delivery systems only allow a multiple choice answer especially for graphical based questions, such as timing diagrams. Finally, when it comes to timing diagrams, making sure the student understands the process in obtaining the answer is more important than the correct answer. To ascertain the student's understanding of timing diagrams, multiple choice answers are insufficient. It should be possible, however, to create a computer script that will not only generate technically correct, random inputs for the circuits in question, but also be able to generate the correct answer via software logic, thus giving a random question and the ability to compare the student's answer to the correct answer. It should also be possible for students to type in a timing diagram using a simple HTML form so that it may be checked against the correct answer. The following is a summary of one effort to create such a system.

[1] Electronics, Computer and Information Technology, North Carolina Agricultural and Technical State University, dekohn@ncat.edu

TIMING DIAGRAM APPROACH

Since the objective was to create a script that could be used for timing diagrams, the first obstacle to overcome was how to display and input timing diagrams via a webpage.

The first approach considered was to use graphical blocks to represent the possible states in timing diagram, similar to the approach used by the simulation software available from Xilinx. Each block would represent a signal portion of equal time and would be drawn so that the blocks could be connected to form the full timing diagram. Displaying the waveforms generated would not be a problem because each section of the waveform would be randomized and then displayed one block at a time. The downfall of this method is how to allow a student to generate a waveform interactively via these basic graphical blocks. A client-side script could be used to allow the student to input their waveform. This approach was abandoned when it was realized that it would probably be a cumbersome interface for the user and that a client side JAVA script would be needed for the implementation (which is beyond the expertise of the author).

The second approach was to use an HTML form textbox to allow the display and input of the timing diagrams. Using the minus sign for a high and an underscore as a low it would be very easy to display and input timing diagrams. This seemed like a straight forward approach that would be easy to implement. Upon testing it was discovered that the font used by default for an HTML textbox was not monospaced (monospaced – each character having the same width). For timing diagrams this would make the method unusable since the inputs and outputs would have to be aligned in the x-axis (time) for them to be a meaningful. Upon investigation, it was discovered that an HTML- textarea font was, by default, monospaced, so that HTML feature was used in place of a form textbox thus solving the issue with display, user input and spacing for the time axis.

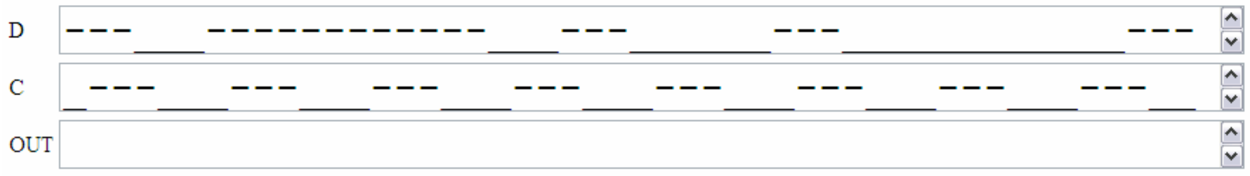


Figure 1 - Example of a Timing Diagram Produced using a TextArea

WAVE-GEN LIBRARY

Now, with the method of displaying and entering the timing diagrams being established, attention was turned to how to produce a "random" waveform that was valid for the various flip flops and latches. Since the goal was to create a system that could be used for multiple assignments, a library of functions would be needed to generate waveforms, verify that the waveforms produced were valid for the gate, flip-flop or latch specified, and emulate the behavior to produce the correct answer to be compared to the student's answer.

The functions for basic logic gates were attempted first. These functions posed no significant challenge since PHP, like many computer programming languages, has functions for AND, OR and NOT. The only challenge was creating routines that would go back and forth from a generated waveform represented by a string (as in Figure 1) and a waveform that was represented by a numeric array representing highs and lows that could be used by the basic logic functions. But this problem was easily overcome.

The challenge came when moving from the basic logic gates to the set-reset flip-flop (SR). The SR flip-flop does not allow both the S and R inputs to be active at the same time. It was therefore necessary to check that the generated waveforms met this criterion. Also, as with any sequential device, to determine the current output, the last state of the output must be known. These problems were overcome and the function was tested, and then a slight modification made so that the SR inputs could be selected as either active high or active low.

Additional functions shortly followed for the Gated SR Latch, D Latch, edge triggered D Flip Flops and other common devices, each generating the input and output waveforms.

In addition to the above timing diagrams, functions were added to do numerical conversions [1] from one base to another, so that the library could also be used for Number System worksheets since this topic is typically covered in digital logic classes.

SYSTEM SCRIPTS

Since most online assessment systems follow a similar pattern of login, assignment display, answer entry and submission/grading, it was obvious that many of these tasks would be able to be handled with the same script for every assignment. A series of scripts were created to handle each of these tasks.

Login Script

The login script would allow the student to enter their name and last 4 digits of their student ID and select the desired assignment.

The student ID is not used as a password per se, but instead it is used as the random number seed for the system. Using the student ID in this way allows a student to work on the assignment off-line and return to the same random assignment when they are ready to submit it for grading.

The student name is used to generate a unique file name for the finished assignment so that it can be reviewed by the instructor of the course.

Setup Script

When the login is complete, a small setup script is executed that sets up the PHP SESSION, variables used throughout the rest of the scripts (listed below) and sets the random number seed. If the student ID is left blank, a random assignment will be generated, but will not be saved, allowing students to use the scripts as a study aid.

Assignment Scripts

Each assignment is made up of two files. The first file generates the “random” waveforms and correct answers. These values are then passed to the 2nd script that displays the webpage to the student.

The actual web page is used twice, first, the setup script sets a variable to indicate that the script should display blank text areas where the student can fill in their answer. Once submitted it is graded and then this webpage script is re-generated to display the student’s answer and whether it is correct or not. At this point it also saves a copy of the HTML page to a password protected directory so that it can be reviewed by the instructor.

Grade Script

This script actually compares the students answer to the correct answers and tallies the score then returns to the webpage to display the results.

Exit Script

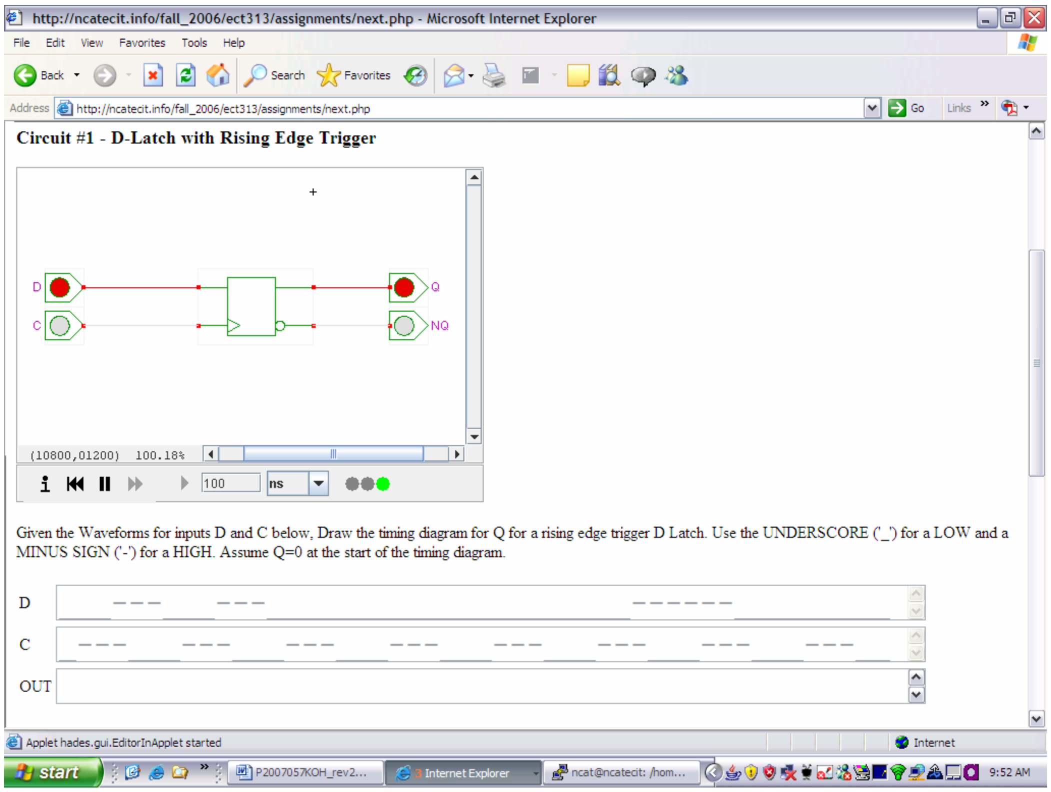
Closes out the PHP SESSION and destroys all the session variables.

ASSIGNMENT CREATION

As stated in the previous section, two scripts are used to create each assignment. The first script uses the wave_gen library described previously to generate each required input waveform and then the correct answer. These are then saved to SESSION variables. These variables are maintained until the session is closed by the exit script or the student closes the browser. Since the grading routines just check for a matching answer, it is possible to add additional questions (e.g. have the students fill out a truth table for the device in question) so long as the correct answer is placed in the correct session variable. This adds quite a bit of flexibility to the system.

The webpage for the assignment contains a combination of PHP and HTML. Since this page is used two times, and also saved to disk, all HTML code is first placed in a variable, and is sent to the browser at one time, instead of line by line. The website itself is not limited in any way, you can use any HTML feature (tables, fonts etc) and choose to add graphics (e.g. circuit diagrams, IEEE symbols for the device, etc.) or even Java applets.

For online students, this system can be coupled with interactive simulations embedded into the webpage. One such simulation system is the Hamburg Design System (HADES) [2]. This Java framework for interactive simulation was created by a team at the University of Hamburg and lead by Norman Hendrich. The system allows for an instructor to create and embed live digital logic simulation into a webpage. With these embedded simulations, the system described within this paper now becomes an interactive, on-line laboratory for digital logic. The students can manipulate the circuit to see how it works then be given a series of timing diagrams to fill out after they have an understanding of the gate, flip-flop or latch within the assignment. These simulations can easily be embedded into the website for the assignments. For more information on HADES see their website at http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/index.html.


http://ncatecit.info/fall_2006/ect313/assignments/next.php - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://ncatecit.info/fall_2006/ect313/assignments/next.php
Go Links
D
C
Q
NQ
(10800,01200) 100.18%
100
ns
Applet hades.gui.EditorInApplet started
Internet
start
P2007057KOH_rev2...
3 Internet Explorer
ncat@ncatecit: /hom...
9:52 AM

Circuit #1 - D-Latch with Rising Edge Trigger

Given the Waveforms for inputs D and C below, Draw the timing diagram for Q for a rising edge trigger D Latch. Use the UNDERSCORE ('_') for a LOW and a MINUS SIGN ('-') for a HIGH. Assume Q=0 at the start of the timing diagram.

```
D    ___---___---_____________________------_________
C    _---___---___---___---___---___---___---___---__
OUT
```

Figure 2 - Embedded HADES Script with Timing Diagram System

FUTURE ENHANCEMENTS

Although templates have been created to aid in the creation of assignments, the level of knowledge in both PHP and HTML required to use the templates is beyond many users who would benefit from the system. Future plans include an online interactive assignment creation tool thus removing this barrier and allowing for widespread implementation of this system.

The script currently uses a minus sign for a high and an underscore for a low and transitions between states are implied but not shown. So far, this has not been an issue with the students who have used the system, but these students have already been exposed to timing diagrams in other courses. The implied transition could be an issue with students new to timing diagrams. Experimentation using a specially defined font has shown promise. It would allow for a graphical block approach discussed above, while still using a font. Issues that have yet to be addressed are: How to force a user to download a font? How to automatically install a font? And how to deal with computers that are "locked" so users cannot install fonts? These issues have yet to be investigated.

CONCLUSIONS

The system has now been in use for about a year and a half and definitely has potential. The students like getting instant feedback on the assignments as well as being able to practice the skills at any time and still obtain feedback. The faculty has praised the system because it cuts down on grading but still provides quick feedback for their students.

A paradigm shift in how students work on these assignments has also been noted. With a non-random assignment students would get together and the weakest student(s) would most likely be given the answer, with little or no explanation. Now that the assignments are random, students still work together, but the stronger students do their own work while showing students who are having difficulty the method not the answer. Then the student who is having difficulty works their own set of problems to reinforce what they were just shown, thus helping the learning cycle.

With the benefits to both the students and instructors the system has proven itself beneficial and development will be continued.

REFERENCES

[1] A.J. Marston, "*A Binary-Octal-Decimal-Hexadecimal-Base36 converter*", http://www.tonymarston.net/php-mysql/converter.html

[2] Norman Hendrich, "*Hades Interactive Simulation Framework*", http://tams-www.informatik.uni-hamburg.de/applets/hades/, Department of Computer Science, University of Hamburg

Daniel Kohn

Daniel Kohn is an instructor of Electronics, Computer and Information Technology at North Carolina Agricultural and Technical State University. He has been teaching for six years and is also pursuing a PhD in Electrical Engineering. He has over fifteen years of industrial experience in the area of computer control and measurement systems.