A Matlab-Based Teaching Tool for Digital Logic

Clark T. Merkel, Mechanical Engineering
Rose-Hulman Institute of Technology

Abstract:
This paper introduces, shows, and makes available a tool which is being used for teaching digital logic devices as part of a course in Mechatronics at Rose-Hulman Institute of Technology. This tool provides a menu driven set of interactive, logic device demonstrations that allows the user to set input states and clock rates to show how a variety of combinational and sequential logic devices behave. This interactive tool is appropriate for use as an in-class teaching demonstration or as an application that can be provided to students to quickly explore common digital logic applications in a laboratory setting. Each application of this menu driven program is written using Matlab program commands and uses a graphical user interface which allows the user to easily control all inputs with the simple click of a mouse button. By changing the inputs of each demonstration and examining the outputs, intermediate states, and truth tables, it is easy for students to begin to understand how common digital devices function.

This program currently demonstrates eighteen different logic gates and circuits. These include logical AND, OR, XOR, NOR, and NAND gates as well as more complicated combinational and sequential logical devices like decoders, multiplexers, flip-flops, latches, counters, LED displays, and drivers. More logical circuit devices are in the process of being added, and these logic demonstrations are available for free distribution at the web address given at then end of the paper.

Introduction:

Digital logic is part of the content covered in the mechatronics course which is taught as a required course in the mechanical engineering curriculum at Rose-Hulman Institute of Technology. For most of the mechanical engineering students who enroll in this class, this course gives them their first exposure to digital electronics, Boolean logic, microcontrollers, and assembly language programming, as well as other selected topics. The course is taught during a 10 week quarter with three 1-hour lectures and one 3-hour laboratory each week. The laboratory content of the course is dominated by learning how to use the Handy Board microcontroller and a variety of sensors and actuators. The laboratory sessions are currently devoted to hands-on exercises that provide them with experience using different sensors and controlling several types of output device with the microcontroller. The students complete six or seven weeks of canned lab exercises to acquaint themselves with the programming skills and capabilities of the microcontroller and sensors. They spend three to four weeks designing, programming, and building a project that requires the microcontroller be used to sense, control, and respond to some design problem of the students' choosing. The project is completed using teams of two

students working together. A wide variety of Lego parts are available to the class, and most students use these for the structural components of their designs, although students often choose use of other construction materials. Currently the use of the lab time is packed pretty full with this content related to working with the microcontroller and project, and it is wished to continue to retain this content.

Understanding digital logic is one of the topics which is covered during the lecture portion of the class. In an attempt to introduce a more interactive method of working with digital logic gate, the set of logic demonstrations has been developed and made available to the students. While this topic could arguably be more appropriately presented by use of digital ICs and bread-boarding the digital experiments such as those of Gasperini [1], experience has shown bread-boarding the circuits to be time and labor intensive and quickly uses up the short of amount of time allocated to cover this material in the class. For this class a better option has been to provide this set of interactive modules which demonstrate the behavior of common digital circuits, but do not require the students spend the time breadboarding and testing the circuits. Instead the student have been given an interactive tool with which they may quickly examine behavior of common digital devices. This provides a quick and convenient tool to discover how variations in input affect the digital output of the circuits.

The logic demonstration programs are written using Matlab program code. The rational for use of this language is that students at Rose-Hulman have been taught use of Matlab as their primary programming language for numerical methods and all students have a license to use this application on their laptop computers. By providing the code in this language, it demonstrates an extension to their current programming skills by showing how Matlab can also be used to create simple computer graphics and incorporate graphic user interfaces. Students are free to examine the open code and enhance their Matlab skills further, although few choose to dig this deeply. Additionally Matlab and Simulink are both used in the class when the topic of system modeling is covered. However, Simulink proved to be neither an easy nor clear tool for developing these hands-on digital simulations. While it is understood that Matlab is not the best graphic development system available, within the context of the course it does an adequate and convenient job.

Digital Logic Demonstrations:

A listing of the eighteen currently offered digital logic gate and circuit demonstrations is given on Table 1. Each demonstration can be run as a stand alone program, or all demonstrations can be run from the program file, **logicdemo.m** which provides a menu driven selection interface to run each of the programs one at a time or concurrently. To give the reader an idea of the type of demonstrations contained in these demos, five of the demos will be briefly presented and described in this paper. If the reader would like to review others of these demonstrations, they are free to download the files from the website given at end of the paper. To run the **logicdemo.m** program you should download all of the files. If you have interest in only a single demonstration file, then only that single file will be necessary to run the demo.

Table 1: Digital Logic Demonstration Program Files

Demo file	Short description
AndGate.m	A logical 2-input AND gate with optional truth table display.
OrGate.m	A logical 2-input OR gate with optional truth table display.
XOrGate.m	A logical 2-input XOR gate with optional truth table display.
Inverter.m	A logical NOT or inverter.
NandGate.m	A logical 2-input NAND gate with optional truth table display.
NorGate.m	A logical 2-input NOR gate with optional truth table display.
NorFlipFlop.m	A simple RS flip-flop based on NOR gates. A change of output state is triggered when one of the two inputs undergoes a Low-to-High transition.
NandFlipFlop.m	A simple RS-flip-flop based on NAND gates. A change of output state is triggered when one of the two inputs undergoes a High-to-Low transition.
DLatch.m	A D-style latch with four inputs: D input, Enable, Set, and Clear. The latch is enabled when the Enable is held at High state. The Set and Clear are used to preset the output state.
DFlipFlop.m	A D-style flip flop with four inputs: D input, Clock, Set, and Clear. The flip-flop is activated by a Low-to-High transition of the Clock. The Set and Clear are used to preset the output state.
JKFlipFlop.m	A JK-style flip-flop with five inputs: J input, K input, Clock, Set, and Clear. The flip-flop is activated when the Clock undergoes a High-to-Low transition. Set and Clear are used to preset the output state.
MemRegister.m	A 4-bit memory register built with D flip-flops used to demonstrate a simple memory circuit. Each bit may be controlled by a button. Output of each bit is updated when the clock undergoes a Low-to-High transition.
RecirculatingData.m	A 4-bit memory register built with D flip-flops which is used shift the output states in a recirculating pattern. Output of each bit is shifted when the clock undergoes a Low-to-High transition. Set and Clear are used to preset individual outputs.
AsyncCount.m	An asynchronous 4-bit counter which displays the value as a hexadecimal value on a seven segment display.
Decoder.m	A 2-bit decoder used to select one of four output lines.
Mplexer.m	A 2-bit multiplexer which passes one of four selected inputs to the output.
Demplexer.m4	A 2-bit demultiplexer which passes the input to one of four selected outputs.
SevenSegment.m	Shows the results of a LCD to 7-segment decoder and display.
LogicDemo.m	A program which runs a selection menu from which each of the above demonstrations can be run. To run this program, each of the other programs must also exist in the current folder.

AND Gate Demo:
Running the **AndGate.m** program opens up the demonstration window for the AND gate shown in Figure 1. The user may toggle each of the inputs between Low and High states and examine the output state. High and Low states are color coded and labeled.
The visibility of the truth table may be toggled on and off.

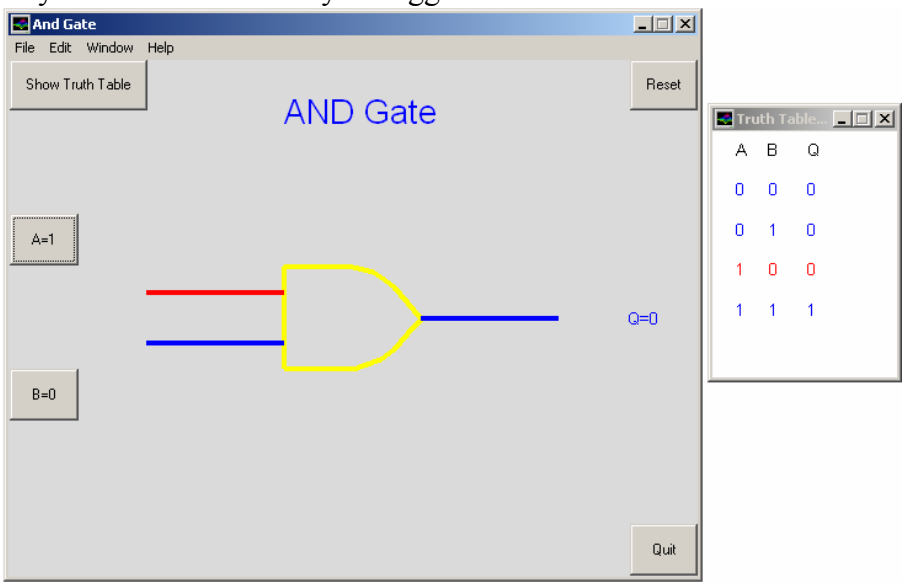

Figure 1: And Gate Demonstration Window with Truth Table

RS Flip-Flop (NOR Gate Flip-Flop) Demo:
A simple flip-flop constructed from two NOR gates is demonstrated by running the **NorFlipFlop.m** program. The demonstration in Figure 2 shows how the output states will flip if the correct Low input state is changed to High while the other input is held Low. For two High input states, the output state is unstable and the demo indicates this result.

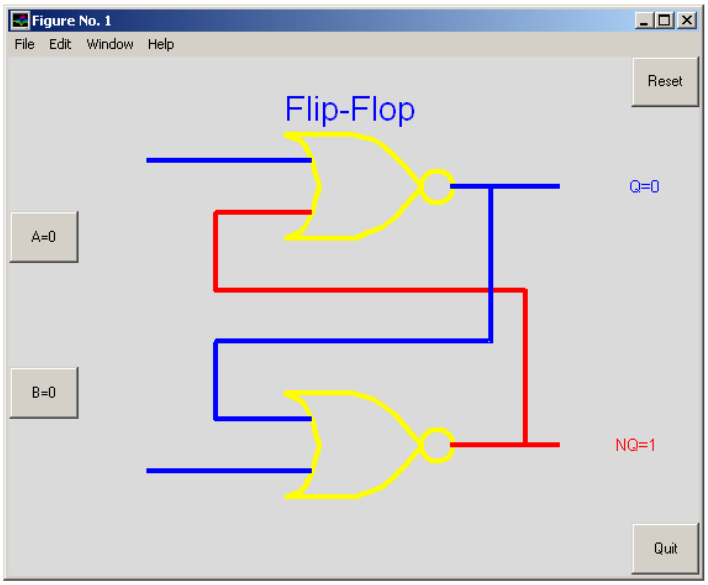

Figure 2: RS Flip Flop Demonstration Window

JK Flip-Flop Demo:
The JK flip-flop demonstration window is shown in Figure 3. This demo includes a clock signal which may be toggled manually or may be made to run automatically. The clock period may be adjusted only over a very small range of values. The Set and Clear buttons are used to preset the output states. The flip-flop is active when both Set and Clear are held at logic level High. Students can see the effect on the output states by changing the J and K inputs while the clock runs.

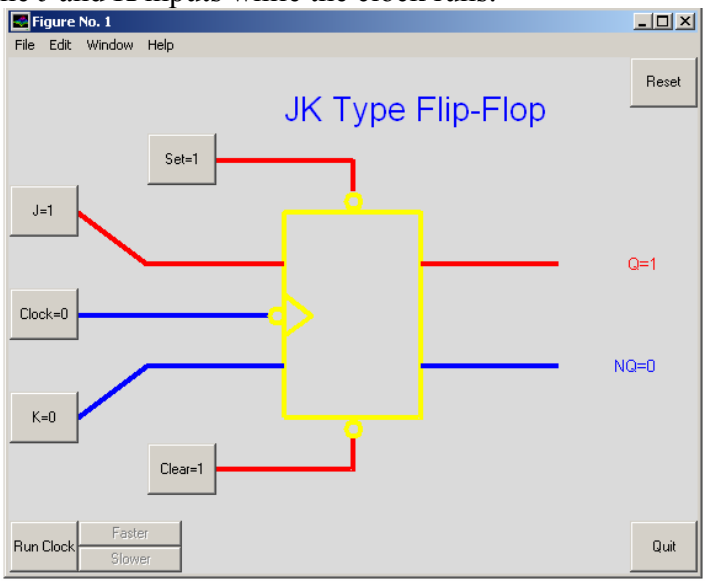

Figure 3: JK Flip Flop Demonstration Window

Multiplexer Demo:
The 2-bit multiplexer demo window run by program **mplexer.m** is shown in Figure 4. It shows how selecting the proper states of inputs A and B allow only one of the four switched inputs to pass through to the output.

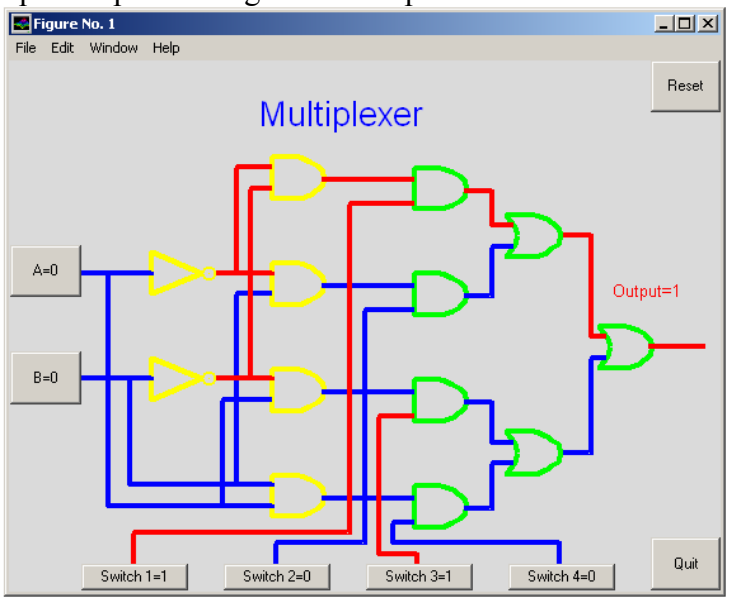

Figure 4: Multiplexer Demonstration Window.

Seven Segment Driver and Display Demo:
This demonstration shows the effect of a BCD to 7-segment encoder/driver and the form of the output on the 7-segment display. The demo may also be toggled to show the selection of the encoded segment lines to display the 4-bit input as a hexadecimal digit.

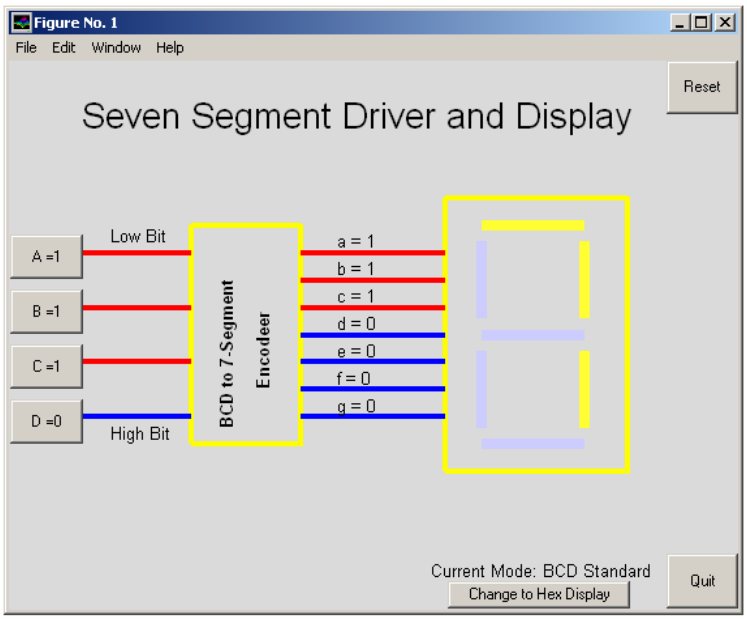

Figure 5: Seven Segment Driver and Display Demo Window

The code for each of these five demonstrations, as well as, all other demos given in Table 1 is free for use and modification at the web address:
http://www.rose-hulman.edu/~merkel/WpClassesME430.htm
The author encourages users who have comments or suggestions on ways to improve the demos or on possible bugs they may encounter in the code are encouraged to provide feedback to the author.

Conclusion:
The digital logic demonstrations presented in this paper are currently being used to introduce mechanical engineering students to common digital logic devices as part of a mechatronics class. Currently the tools are given to the students to freely experiment with and they serve more as a secondary learning tool or reinforcement activity. The tools are also used for demonstration of logic devices during lecture. There are plans to use these tools in a more structured learning exercise or tutorial, but this has yet to be completed. Student qualitative comments indicate that they find these tools useful to help them understand the behavior of common digital logic devices. Whether used with an overhead display and presented as part of a lecture, or given to students to use for their own interactive exploration, these demonstrations one more tool available to quickly and easily explain the behavior of digital logic circuits.

Bibliography:

1. Gasparini, Richard E.,"Digital Experiments", pub. by Hayden Book Company.
2. Greenfield, Joseph, "Practical Digital Design Using ICs", pub. by J. Wiley & Sons.
3. Pratap, Rudra, "Getting Started with Matlab, A Quick Introduction for Scientists and Engineers", pub. by Oxford Press.
4. The Math Works Inc, "Matlab User's Guide", pub. by Prentice Hall.
5. Bolton, W., "Mechatronics, Electronic Control Systems in Mechanical and Electrical Engineering", pub. by Addison Wesley Longman Limited.