

AC 2007-515: CLASS PROJECTS WITH GRAPHIC USER INTERFACES IN MATLAB

Min-Sung Koh, Eastern Washington University

Esteban Rodriguez-Marek, Eastern Washington University

Claudio Talarico, Eastern Washington University

Class Projects with GUIs in Matlab

Min-Sung Koh, Esteban Rodriguez-Marek, and Claudio Talarico

School of Computing and Engineering Sciences
Eastern Washington University
Cheney, Washington 99004
USA

Email: {mkoh, erodriguezma, ctalarico}@ewu.edu

Abstract

Graphical user interfaces (GUIs) have facilitated the work of engineers in using software packages. Simulation of various engineering models becomes much easier when modifying a variable in a GUI rather than running the program from a command prompt. Typically GUIs have been implemented in languages such as C/C++ and/or JAVA. In line with this trend, these languages have been part of the Electrical Engineering (EE) curriculum at Eastern Washington University (EWU). However, the growth of Matlab in the recent years also makes it imperative for it to be embedded in different classes throughout the curriculum. Matlab is currently not only used as a mathematical manipulation tool, but also for the design and simulation of various dynamic systems. Similarly, hardware description languages (HDL) are ubiquitous in the design and simulation of integrated circuits. When combined, HDL and Matlab make a powerful team for designing, simulating, and implementing digital circuits into field programmable gate arrays (FPGA). This industry trend is the driving force behind the EE curriculum revision at Eastern. While a C/C++ class is still offered, JAVA programming is no longer a mandatory course. To balance this issue, the use of Matlab and HDL has been increased significantly. One consequence of this change was that the students' ability to design GUI interfaces became severely diminished. Fortunately, Matlab also provides the capability to program GUI interfaces. This paper presents a set of digital signal processing (DSP) student class projects that include the design of GUI interfaces for simulation and testing of systems entirely through the use of Matlab. The paper also presents a preview of follow-up lab developments which will include the implementation of complete systems into a DSP board using Matlab. In addition, there are classes under development in which the projects will require downloading Matlab algorithms into FPGAs. This paper's overarching goal is to demonstrate that EE curricula do not need the teaching of many different high-level programming languages if Matlab is taught vertically throughout the curriculum.

1. Introduction

Typical Electrical Engineering (EE) curricula have mandatory software and hardware components, in fulfillment of the criteria of the Engineering Accreditation Commission of the Accreditation Board for Engineering and Technology (EAC of ABET) [1]. This is congruent with the effort to allow students to study systems which can be simulated and controlled through software. The particular curriculum at EWU included 11 quarter credits classes in high level programming languages such as C/C++ and/or Java. A summary review of the curriculum revealed that students graduated with not only with knowledge of those languages, but of assembly, HDL, Matlab and Pspice, which are naturally embedded into the relevant courses, such

as *Signals and Systems*, *Computer Architecture*, *Electronics*, etc. *Microcontroller Systems* embeds into its 4 credits a significant portion of assembly language as a mandatory class. One of the electives in the program is an advanced HDL class. Under this setup, students would take between 15–19 credits with large programming components. With the ever enlarging scope of information students have to obtain in their limited 4-year college career, the large number of credits dedicated to programming was limiting the space for core or elective EE courses. Due to this limitation the 11 credits of Java programming were changed into a more directly related 4-credit *C for Engineers* course. One of the concerns that arose with this modification was the elimination of GUI programming, a major subject in one of the Java courses deleted from the EE curriculum. To address this concern, GUIs were incorporated for in a lab experiment using Matlab. The reason behind this was to present one more tool for the student to introduce a simple way of manipulating parameters in a given design. This paper contends that the students' ability to design and use GUI systems was not harmed by the elimination of Java from the EE curricula. Further, it can be argued that GUI programming can be very beneficial to EE students. The paper is organized as follows. Section 2 presents a brief summary of the GUI projects students presented in a DSP class, as well as students' evaluation of the process. Based on the pedagogical observation from the projects, further curriculum revision under development are explained in Section 3, namely, to tie Matlab programming skills with hardware implementation in DSP and/or FPGA boards.

2. GUI system designs using Matlab

Digital signal processing is taught as a senior-level, 4-hour/week lecture, 2-hour/week laboratory class. The lab culminated in an intensive team-oriented class project. The projects were proposed by the students to the instructor, who revised the project for appropriate content and size prior to approval. All specifications were chosen by students themselves, save for one criterion: the inclusion of a Matlab-based GUI system. Most of the students are seniors in EE, and more than 60% of them had prior experience in programming in a high-level language such as C/C++ or Java. It is worth noting that 20% of the students did *not* respond to the survey, and only 20% of them indicating *not* having taken a programming class yet. A complete summary of the survey data is shown in section 2.3. A necessary prerequisite for the DSP class is the second quarter of *Signals and Systems*, which implies that all students had had at least two quarters of experience in using Matlab. It is worth mentioning that most classes in the curriculum have a laboratory component¹. Currently there is no separate class only on Matlab programming, but the first two labs on *Signals and Systems I* are dedicated to introduce Matlab to the students. The rest of the labs in that class are also done in Matlab, but are directly geared towards enhancing students' understanding of the concepts learned in class, such as Laplace, Fourier, and Z transforms, etc. The use of hands-on hardware related to signals and systems is incorporated in *Circuits I and II*, a prerequisite for *Signals and Systems I*. In the DSP class, only one lab is dedicated to a brief tutorial on how to make a GUI system in Matlab. The remaining labs are typical DSP labs, such as filter design for both finite impulse response (FIR) and infinite impulse response (IIR) filters, etc. In the two-hour intro to GUI systems, a reduced version of the handout given to students is shown next:

“The tool to design a GUI system can be called by typing “>> guide” in the Matlab main command window. This will give us the “Create Quick Start” window shown in Figure 1.

¹ This is not entirely true, as the laboratory accompanying the class on Digital Logic is a class on its own. Also, the class on Electromagnetism is cross-listed with the Physics Department and, thus, does not have a laboratory component.

Choose “Blank GUI (Default)” to make the desired new GUI system and click “OK”. The window in Figure 2 will be invoked to design your own GUI system. On the window in Figure 2, you may design any kind of GUI system by clicking and dragging from the menu. Possible items include pop-up menus, push buttons, editing boxes, check boxes, figures, etc. Once an appropriate menu in Figure 2 is chosen with the mouse, it can be put anywhere in the GUI system with a simple mouse click. Examples are shown in Figure 3. After putting each menu, double clicking in a menu, a “property inspector” window will pop up. This window allows manipulation of the properties and characteristics of each item such as string, tag, value, background color etc. Saving the GUI system shown in Figure 3 as, say, GUIexample (or whatever you want) will result in Matlab saving the file as “GUIexample.fig”. Matlab will also generate automatically the file “GUIexample.m”, associated with the previous file. Typing “GUIexample” in the Matlab command line will invoke the GUI you created. All actions taken when each menu is chosen can be added into the “GUIexample.m” file, which is automatically generated by your design, “GUIexample.fig”, in Figure 3. In the automatically generated “GUIexample.m” file, many functions corresponding to each menu will be shown. These functions must be revised to incorporate actions, which will happen when each menu is selected by a user.”

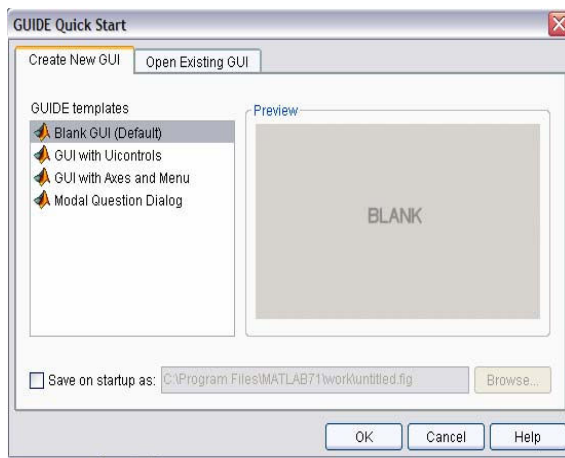


Figure 1. Starting window to design a GUI system

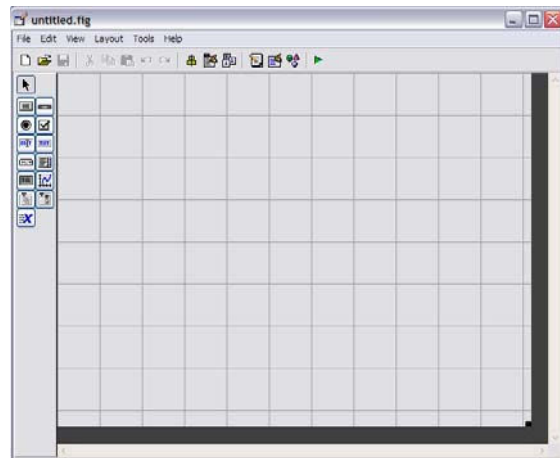


Figure 2. Window to design each component in a GUI system

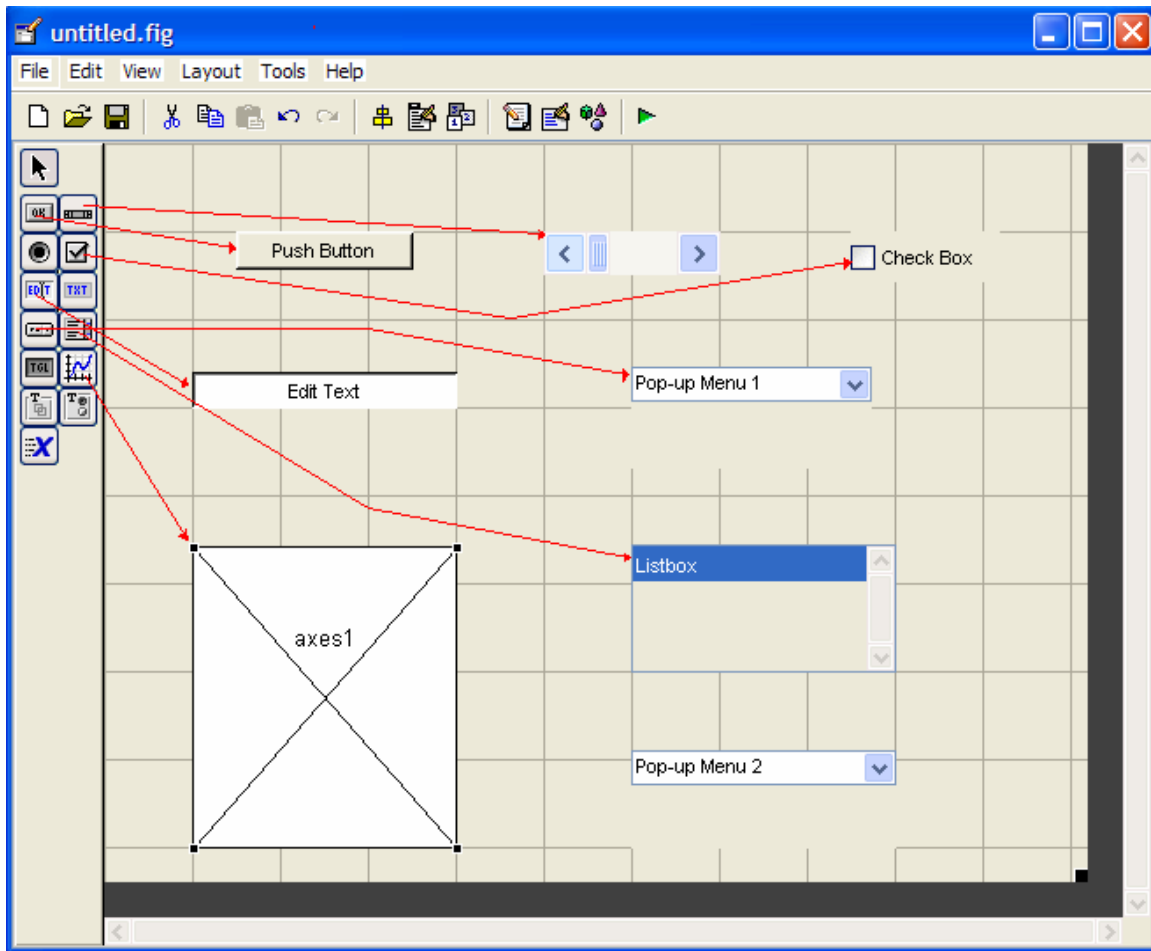


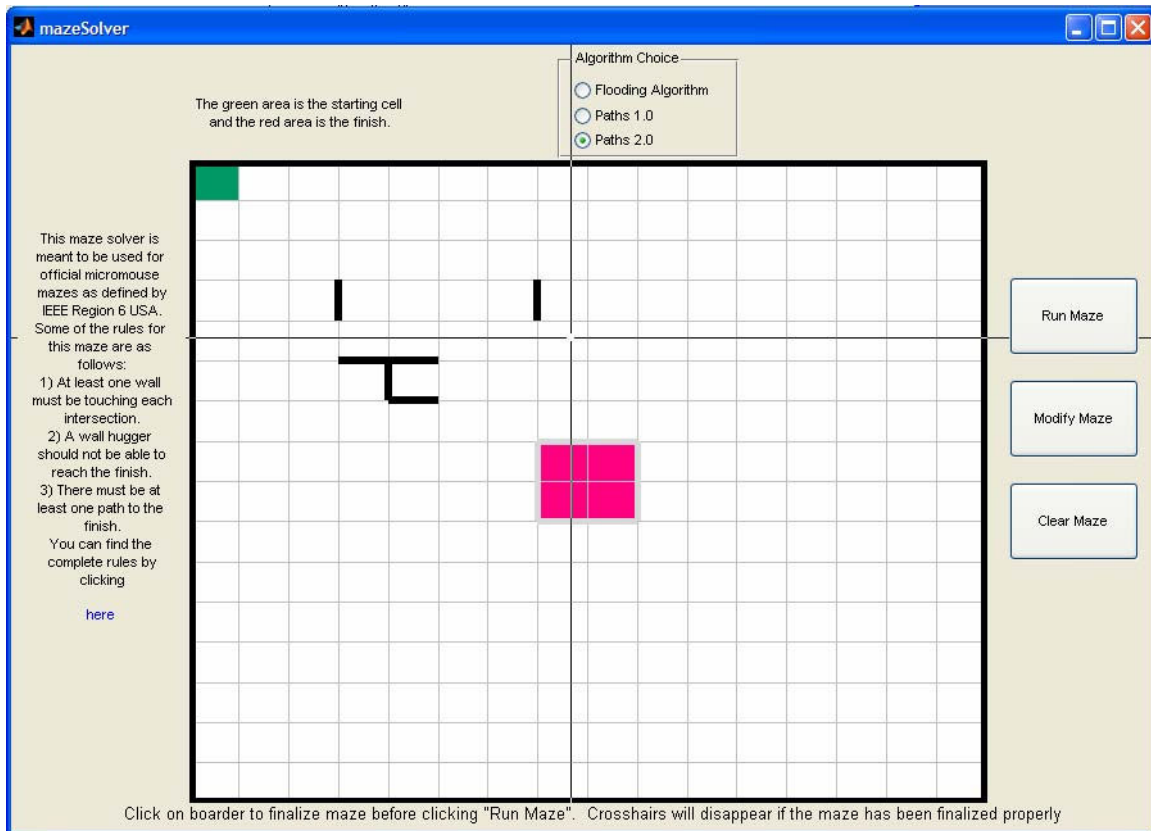
Figure 3. An example of components in a Matlab GUI system

Based on the instructions provided above, students were encouraged to learn more about GUIs throughout the quarter, with the prior understanding that it would be a mandatory portion in their final projects. Class projects of two groups are described in the Sections 2.1 and 2.2, respectively. It was observed that the learning curve for designing GUIs with Matlab is not steep, even though the initial attempts did indeed appear to be time-consuming for the students. Note that all graphs in Figure 4 and 5 are obtained by running the students' class project Matlab code.

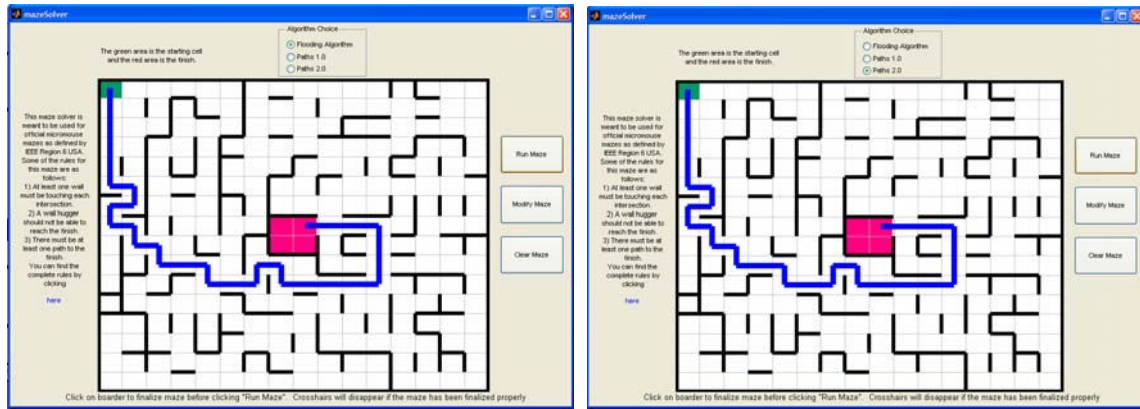
2.1 Example 1 for GUI project

The first of the two examples presented in this paper consisted of a group of 3 students, who chose to implement an algorithm to find the shortest path in an arbitrary maze of 16x16 blocks. Note that this is similar as that of the popular micromouse competition [7]. To make it more challenging, students had to design their own algorithm to find the shortest path, and then compare it with the well-known Bellman flooding algorithm, or *flooding* for short. Despite the amount of real signal processing involved is minimal, the project was approved by the instructor to encourage the team in their efforts to build the first micromouse robot at EWU. Students designed a GUI system both to simulate their algorithm and to compare it with the flooding algorithm. In the designed GUI system, all walls in an arbitrary maze can be installed by clicking a mouse button at the grid lines, as shown in Figure 4(a). As long as the mouse moves, the crossing line shown in the rectangular window moves as well, to let the user know the position where a wall is about to be located. The building process to make a maze is continued

until one of the boundaries of the maze, denoted by a black rectangular, window is clicked. The GUI system shown in Figure 4 consists of simple GUI menus such as radio buttons and push buttons. The radio buttons are used to select the flooding algorithm or the students' two different algorithms. Two of the push buttons allow either to make a new maze or to modify an existing one, respectively. The third one runs the algorithm selected by the radio buttons to find the shortest path in the user-designed maze. The green block indicates the starting point and the pink block shows the finish line. The rectangular window for a maze is composed of 16x16 blocks, as defined by the IEEE Region 6, USA, for the micromouse competition [7]. The results of two different algorithms are shown in Figures 4(b) and 4(c) with blue lines. Figure 4(b) is the result of the flooding algorithm and Figure 4(c) is the result of students' algorithms. An unexpected side result of this project is the student-designed search algorithm, which shows promise in terms of complexity when compared to the flooding algorithm. Further analysis for computational complexity is currently under investigation. To make the GUI interface for their projects, students investigated the capabilities provided by Matlab to implement their idea. Even though several problems (e.g. active X controls) arose during implementation, the use of Matlab as a tool for designing a GUI did not present a barrier to implementing their ideas.



(a) Figure showing how to put walls by clicking a mouse button to make an arbitrary maze.



(b) Result by the flooding algorithm

(c) Result by students' algorithm

Figure 4. Example 1 to design a GUI system with Matlab

2.2 Example 2 for GUI project

The second example presented in this paper is a simulation tool for the design of various IIR and FIR filters, a main topic in the class. The project was implemented with a GUI system programmed to obtain several parameters needed for filter design. The GUI interface is shown in Figure 5. Either IIR or FIR filters can be selected in the top radio button group. The IIR radio button has a pop-up menu to select from several IIR filters, such as Butterworth, Elliptic, Chebychev, etc. The FIR filter radio button has a pop-up menu to select from either lowpass, highpass, bandpass, or bandstop filters. The remaining boxes are to obtain design parameters for a filter such as stopband, passband, passband ripple, stopband ripple, etc. Once those parameters are input through menu components in the GUI system, the application starts finding filter coefficients by clicking the “Apply” button. The “Close” button closes the GUI window. Once the “Apply” button is pushed, the filter coefficients for the desired parameters are shown in list boxes denoted by “a” and “b”. In addition, the frequency response of the newly designed filter is shown in two figure windows denoted by “Magnitude of filters” and “Phase angle of filter”. It is fair to point out that a few mistakes were noted in the simulations that led the instructor to quickly find out where the errors were in the code.

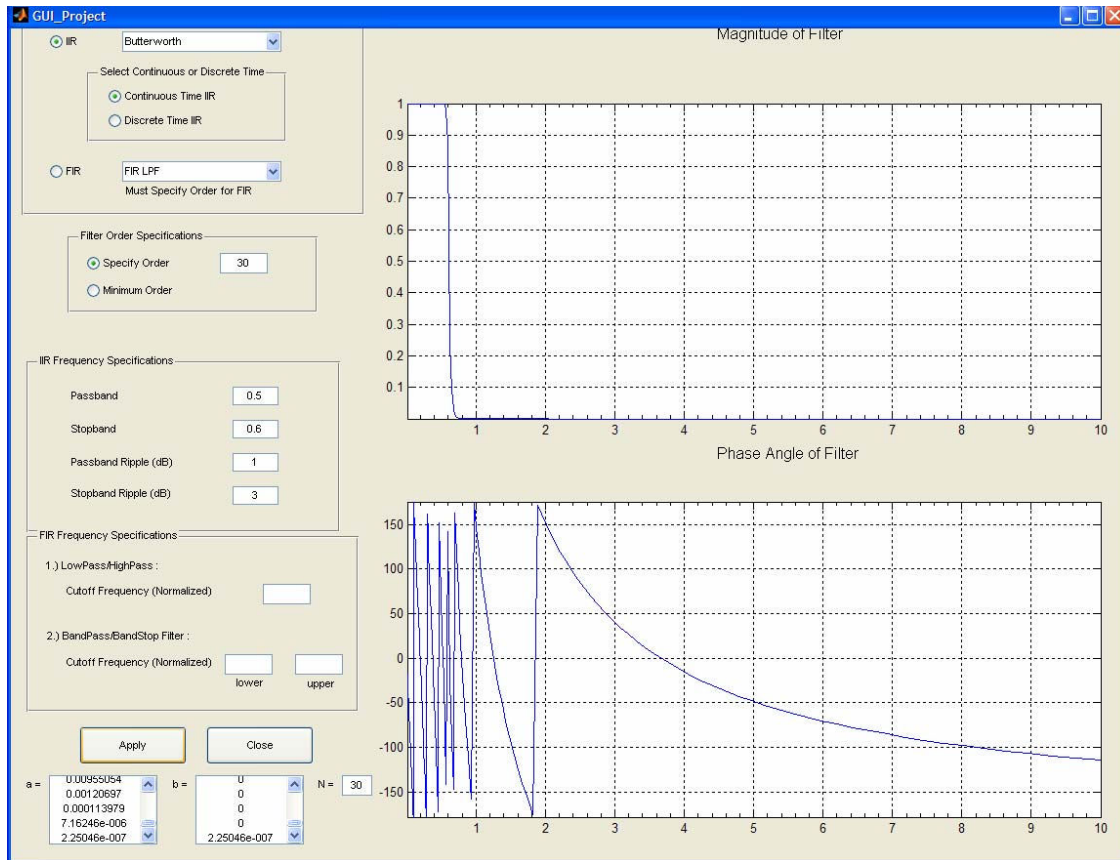


Figure 5. Example 2 to design a GUI system with Matlab

2.3 Students' Evaluation about GUI projects

At the end of the class, a survey was performed to have direct feedback about making a GUI system using Matlab. It is important to note that this survey was the only method of assessment done up to this point. This was done to directly evaluate students' perception about their ability to design GUIs without the use of Java. All results of the survey are shown in Figures 6 and 7. However, note that these results should be taken with a grain of salt, since the sample space is very small (5 students). The survey is composed of total 8 questions, which are shown as follows. Questions from Q2 to Q4 and from Q6 to Q7 are scaled from 1 (very easy) to 10 (very hard). For the question Q5, it is scaled from 1 (Strongly agree) to 10 (Strongly disagree).

- Q1. What computer language classes did you take?
- Q2. How comfortable do you feel programming in C/JAVA/C++?
- Q3. How comfortable do you feel programming in Matlab?
- Q4. Comparing to the computer language classes (e.g., JAVA), what do you think about Matlab in making GUI (Graphic User Interface) software?
- Q5. Do you think that Matlab should replace other high level languages taught in engineering?
- Q6. How do you feel about using Matlab for general engineering programming (not GUI)?
- Q7. How do you feel about making a GUI system using Matlab?
- Q8. What are your comments about making a GUI software using Matlab?

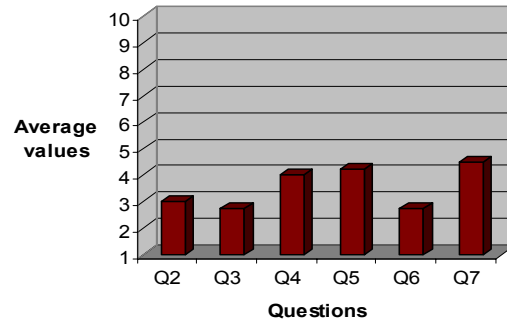
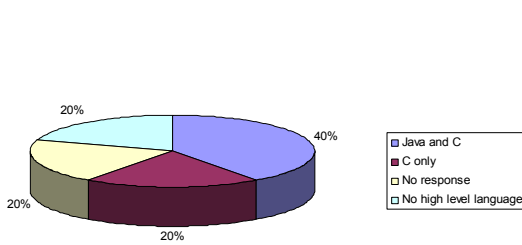


Figure 6. Students' high level language experience. Figure 7. Average values for each question

Figure 6 is the result for the question 1 (Q1) in the survey. The results for other questions from Q2 to Q7 are given in the Figure 7. According to the answer to Q4, students seem to feel a little bit more comfortable designing a GUI system with Matlab than with other high level languages. Even though it was the first time for all of students to use GUI tools in Matlab, students felt that it was not hard to design a GUI system with Matlab, result validated by the answer to Q7. Note that students had only one lab about GUI system design. Hence, we believe that students can learn easily how to design a GUI system with Matlab if more labs are provided. In general, students feel comfortable in using Matlab, which is shown from the answers to questions, Q3 and Q6. In addition, students seem to be roughly neutral to replace high level languages with Matlab. Note, though, that the particular languages were not specified. Since both C and C⁺⁺ are understood by students to be essential for engineers to implement a system in hardware, most of students were hesitant about question Q5. For the last question, Q8, there are several comments such as “*I like it but there are problems that this new GUI system have like active X controls, half of them don't work (aren't compatible). I also think the introduction to GUI could be more involved ~~~*”, “*Yes, helpful to know GUI, which I never know about it and I can learn a new subject (GUI)*”, “*It was intriguing and instructive*”, and “*It would help to go over in lab a few more of the GUI objects, specifically the radio buttons and button groups*”.

3. Course renovation for DSP and/or FPGA implementation using Matlab

This section provides a brief explanation of the on-going plan for course renovations in the EE curriculum at EWU. The overarching goal is to incorporate a method to go from system level design to real implementation of either a logic circuit or a signal processing algorithm. Many EE courses are based on thorough understanding of theory. For instance, some classes, such as DSP, *Digital Communication*, and *Digital Feedback Control*, rely on deep theoretical understanding before undertaking the task of merging theory into practice. Much of the theory in those classes (and others) can be verified in the lab by simulation based on high level languages and/or by actual circuit implementation. One of the most popular ways in academia to do this has been through simulation using Matlab, which already provides many toolboxes tailored for various specific topics. One more advantage of using Matlab is that simulation can be deployed into hardware thanks to recent leaps in device technology. All design processes can now be achieved by a one-step approach, i.e. starting from the original system-level design of an algorithm going directly into its implementation in either a DSP or an FPGA chip. Hence, it is only natural to adopt these approaches in the classroom. This pedagogical renovation is already shown in other universities, such as in [2] and [8], to name a few. In [2], a software-defined radio was implemented through the one-step approach using Matlab and a Lyrtech board [5]. The Lyrtech board has DSP and FPGA chips with other peripheral devices such as analog-to-digital converter

(ADC) and digital-to-analog converter (DAC), etc. Since many algorithms in traditional EE classes can be implemented into hardware in lab environment using Matlab, a few EE courses in EWU (e.g., DSP and Digital Communications) are under renovation following the current trends. The remaining of this section will provide a discussion of the lab renovations to do one-step design from high level into hardware implementation.

As shown in Figure 8, a system design at high level using Matlab can be implemented into DSP and/or FPGA boards with two different mediating programs of Code Composer Studio (CCS) for C/C++ and SignalGenerator/ISE/ModelSim for HDL codes. At each approach, Matlab needs special toolboxes depending on the choice between DSP and/or FPGA boards [3][4]. A similar approach to do real implementation from Matlab to DSP/FPGA is shown in [2][5].

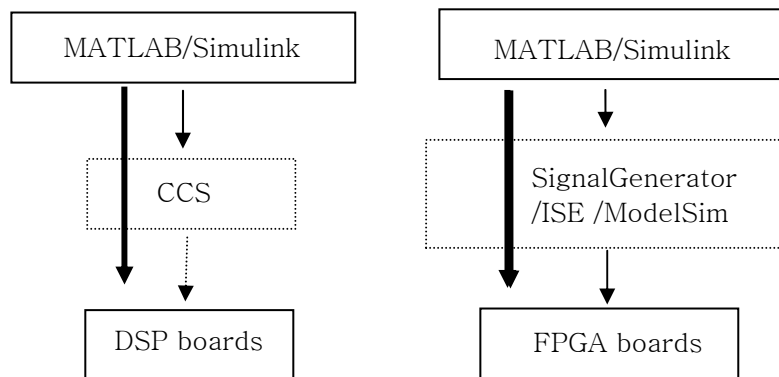


Figure 8. Process to download Matlab codes into a DSP and/or FPGA boards

A brief explanation about the development of labs with the first approach is as follows. In Figure 8, a TI DSP board, TMS320DM642, was chosen for future capstone, senior, and/or research projects. To download the Matlab file into a DSP board, the Simulink toolbox must be available in Matlab, as well as the toolbox for “Embedded Target for TI C6000 DSP”, in which all interface parts such as Audio ADC, Audio DAC, Video Capture, Video Display, etc. for the TMS320DM642 board are available. In addition, the software called CCS 3.1 must be installed to change Matlab code into implementable code for DSP boards. Once the environments are ready, a few simple labs are used to understand the overall procedure to download Matlab code into a DSP board. This process is shown in Figures 9 and 10. Once the Simulink files are obtained, the building process can continue by clicking the “Incremental build” icon in Simulink. This will call the CCS program, generate downloadable codes to a DSP, connects the host computer to a DSP board, and download the changed codes into the DSP board. Thus, the algorithm originally programmed in Matlab is now executed by running the downloaded program on a DSP board. In Figures 9 and 10, note that specific ADC and DAC components for the TMS320DM642 must be used for the DSP board. The system in Figure 9 simply obtains a speech signal from a microphone connected through the “MIC” in the DSP board. In the DSP board, the ADC digitizes the speech signal and the DAC changes the digitized signal into an analog signal, which is further connected into the “line out” in the DSP board. The simple system in Figure 9 gives an idea about how to interface the signal into the DSP board using Matlab. The system in Figure 10 shows how to read signals and plays the signal in the DSP board. A speech signal is read from a file and is played through the DSP board. After verifying the first approach in Figure 8, inexpensive DSP boards, such as the TMS320C6416 (which save for video processing, has similar capabilities than the more expensive TMS320DM642) were chosen for student labs. Lab manuals are under development with the TMS320C6416 board. The lab manuals under development have

similarities to the labs shown in [6], but different because several other applications are under development.

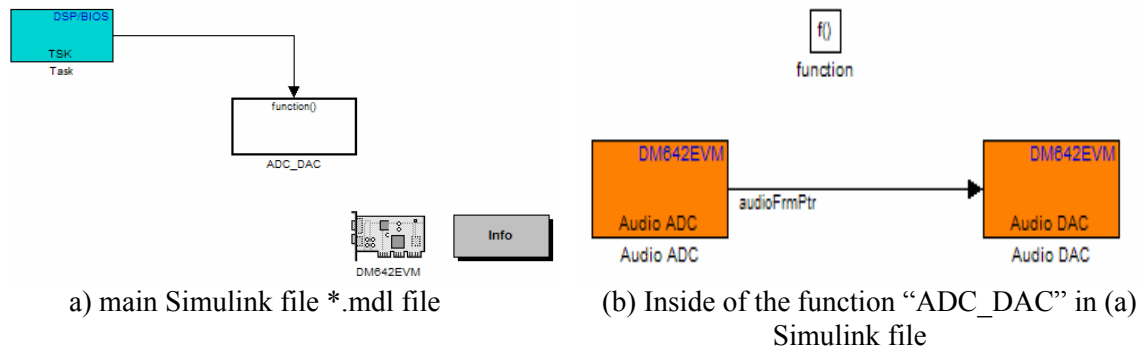


Figure 9. Simple example 1 to download Matlab codes into a DSP board

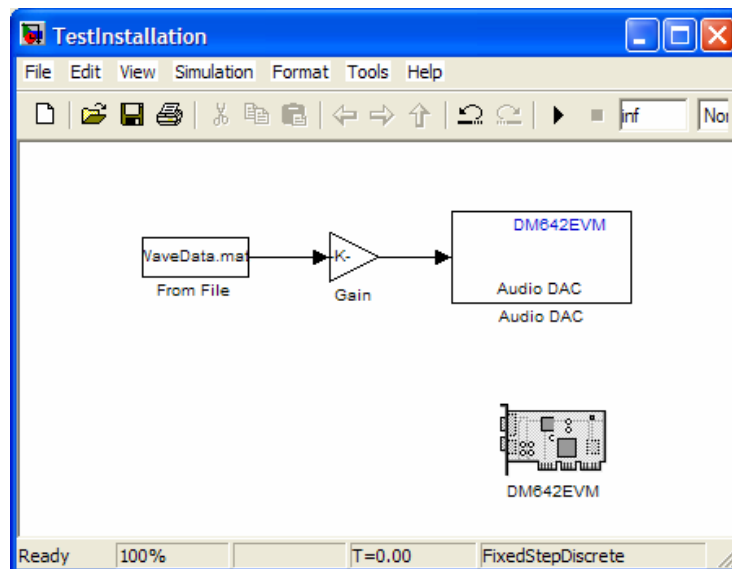


Figure 10. Simple example 2 to download Matlab codes into a DSP board

For the second approach in Figure 8, the Xilinx software such as ISE 8.2i, System Generator 8.2.02, and ModelSim XE III 6.1e are installed. A simple example to understand whole process is given in Figure 11. In Figure 11, a Simulink box (by way of Xilinx Blockset) denoted by MCode is generated by the MATLAB code of “SimpleFun.m”, which is the simple transparent function, $y=x$. The simple MATLAB function is connected with “Gateway In” and “Gateway Out”, which are 4-bit input and output in the FPGA board. Each input and output line is associated with switches and output connectors for inputs and outputs respectively in a FPGA board. After making the Simulink file shown in Figure 11, VHDL codes and an ISE file, used for the Project Navigator of Xilinx, are generated by clicking the “Generate” tab in the System Generator shown in Figure 11. The generated ISE file is opened by running the Project Navigator of Xilinx-ISE 8.2.03i. Using the Xilinx-ISE, the Simulink file shown in Figure 11 is synthesized and downloaded into the FPGA board (a SPARTAN-3E board was used). As a result, it is observed in the FPGA board that all outputs are exactly followed by changing input switches. From the simple example, students can understand the whole process of implementing high level designs

done in MATLAB into an FPGA board. Based on the lab, more intricate labs are under development for a more advanced FPGA board, the Virtex-4. Several examples on how to incorporate high-level design done in MATLAB into FPGAs, and downloading it into hardware for further simulation are explained in detail in [9].

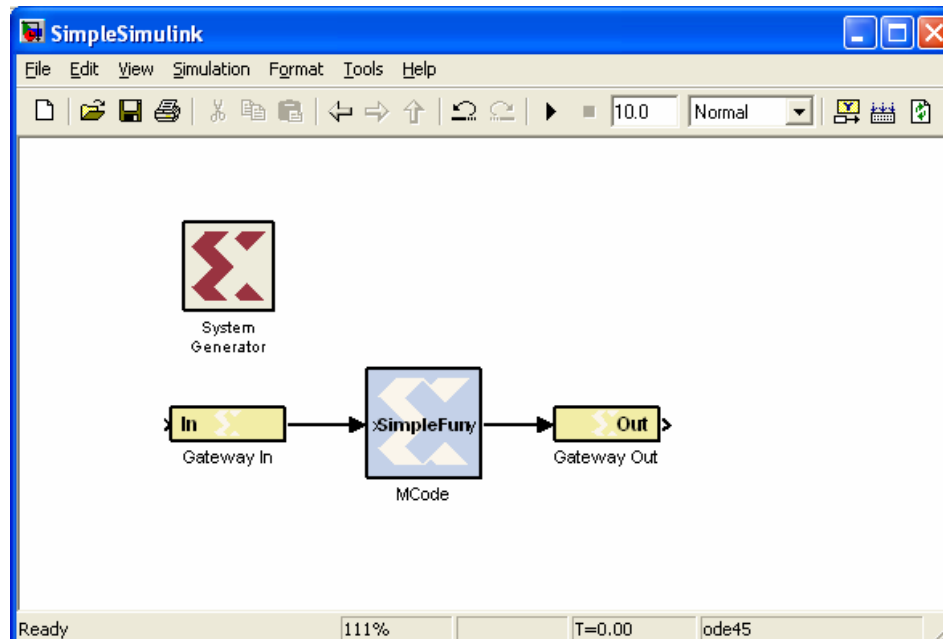


Figure 11. A simple example to implement MATLAB code into a FPGA board

4. Conclusion

This paper shows Matlab as an alternative to high-level languages such as Java for an EE curriculum. The EE program in EWU underwent recently a curriculum renovation which involved removing a sequence of two programming classes, *Java I* and *Java II*, and introducing a new class, *C for Engineers*. With the curriculum space occupied by *Java I* and *II*, more EE elective credits were made available to students and the emphasis on other programming and simulation tools, such as Matlab, HDL, assembly language, and PSPICE, were increased. A concern arising from removing Java programming was the lack of some of the interesting capabilities of the language, such as GUI systems. To show that this weakness can be overcome, GUI system design was added into a class project of one EE class. This paper described the curriculum revision and presented a couple of examples of class projects involving GUI systems developed in Matlab. Observing students' class projects with the added GUI design criteria, it was concluded that programming skills in EE can be achieved without over-emphasizing other additional high level language. In addition, the paper describes further curriculum renovations are under development, which include the translating of high-level designs directly into DSP or FPGA boards. It is worth noting that these changes were triggered by a rigorous continuous improvement plan, developed for upcoming ABET visit. A comprehensive curriculum assessment is in preparation, to assess and evaluate the effects of the curriculum changes.

Acknowledgements

The authors want to express their appreciation to the students who did class projects with the additional GUI criterion. We would like to thank Steven Reynolds, Christopher Nuesmeyer, Viengkham Chindamany, Zachary McMackin, and Christopher Jarvis for submitting all their work. Thanks are also due to the anonymous reviewers, whose comments greatly improved the quality of this paper.

References

- [1] Engineering Accreditation Commission, “Criteria for Accrediting Engineering Programs, Effective fro Evaluations During the 2006-2007 Accreditation Cycle”, ABET, Inc.
- [2] Sven Bilen, “Implementing a hands-on course in software-defined radio”, ASEE annual conference, Chicago, IL, Jun. 2006.
- [3] The MathWorks Inc, “Embedded Target for TI C2000™ DSP 2.1”, <http://www.mathworks.com/products/tic2000/description1.html>
- [4] The Xilinx Inc, “The Mathworks system level design tools and services”, http://www.xilinx.com/products/software/sysgen/mathworks_system.htm
- [5] The Lyrtech Signal Processing, “From Matlab to real-time systems”, http://www.lyrtech.com/DSP-development/dev_tools/index.php
- [6] R. Chassaing, “Digital Signal Processing and Applications with the C6713 and C6416 DSK”, Wiley-Interscience, 2005.
- [7] IEEE Region 6, “Micromouse contest rules, Region 6, Rules for the maze”, <http://www.ewh.ieee.org/r6/lac/students/microm/MMRULES.html>
- [8] B. Dunne, “DSP-Based low-cost digital communications”, ASEE annual conference, Chicago, IL, Jun. 2006.
- [9] The Xilinx Inc, “Xilinx system generator for DSP, version 8.2.02, user guide”, Nov. 28, 2006.