Methodologies to Bring Embedded Systems to Non-EE Students

Shekhar Sharad
National Instruments
11500 N Mopac Expwy, Austin, TX
78759, USA
+1-512-683-5219
shekhar.sharad@ni.com

ABSTRACT
With embedded systems being used in every industry, it is important to empower domain experts who are not embedded design engineers to be able to design, prototype and deploy them in their applications. In this paper, we explore some graphical methodologies available today that provide a higher level of abstraction that can help professors teach embedded systems to non-EE majors. We will identify the advantages such methodologies present and explain with some marquee examples such as ChallengeX and LEGO.

Categories and Subject Descriptors
Dataflow programming, event structures, abstraction

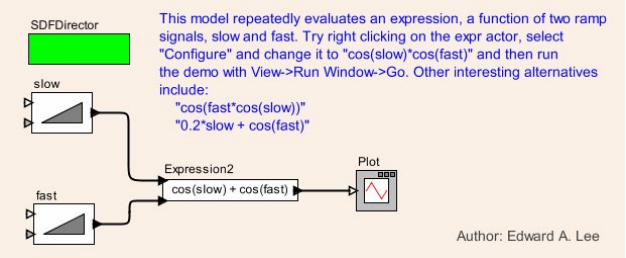
Keywords
Embedded, innovative teaching methodologies, hands-on learning, non-EE majors

1. INTRODUCTION
Embedded systems are becoming increasingly pervasive, from space stations to everyday objects such as cell phones and microwaves. With technologies converging and silicon shrinking, embedded systems now present viable opportunities for individuals and teams that are not embedded design engineers but domain experts in their field of choice. For example, a mechanical engineer building the next generation vehicle may not be an embedded design engineer but is an expert on the dynamics of a vehicle including the design of the engine and the drive-train. Hence there is a need for tools that provide an increasing level of abstraction while providing the power and flexibility of embedded systems. One of the biggest hurdles that traditional programming methodologies present to teaching embedded systems to non-EE majors is that of complexity. Non-EE majors may have little to no textual programming experience and hence there is a need for ways that present an intuitive interface to designing embedded systems so that Professors can explain the concepts and students can understand and use embedded platforms for their projects.

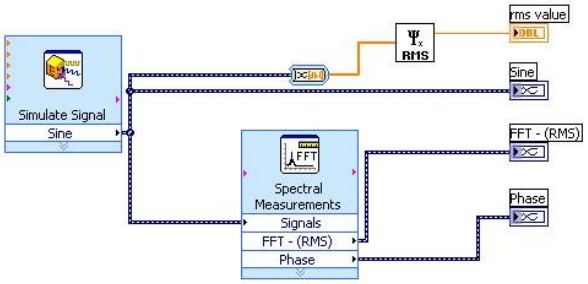
One such effective methodology is using Graphical tools and techniques to teach embedded systems. In this paper, we will present how graphical programming present a viable platform to teach embedded design. We will elucidate on some of the advantages that such graphical programming methodologies present. We will also show how graphical programming methodologies provide a flexible platform that can be used to target embedded hardware such as FPGAs, DSPs or Microcontrollers giving users the option to completely design, prototype and deploy their system. We will also list some of the marquee programs around the world that have been using such techniques to teach embedded systems.

2. GRAPHICAL PROGRAMMING
It is common in embedded and signal processing communities to use block diagrams to represent systems. The blocks typically represent computations and the arrows connecting the blocks represent flow of data. Graphical programming is based on the "dataflow" paradigm which in mathematical terms is a directed graph whose nodes represent computations and the arcs represent streams of data. The nodes may consist of subsystems that consist of a series of nodes and arcs or contain algorithms implemented in a primitive language such as assembly or C. As it can be observed, graphical programming languages lend themselves naturally to embedded and DSP programming which has resulted in a lot of research and development in graphical programming languages.

There are a number of tools that allow a mixture of visual and textual programming such as Signal[1], Lustre[2] and Silage[3]. Completely graphical programming environments such as Ptolemy[4] from University of California, Berkeley and National Instruments LabVIEW[5] have evolved to present powerful tools that professors can use to teach embedded systems without compromising on the flexibility and control that the traditional methodologies provide. Figure 1 shows a typical screenshot from both Ptolemy and LabVIEW.



(a) [4]



(b)

Figure 1. Block Diagrams from Ptolemy (a) and NI LabVIEW (b)

As it can be seen from figure 1, both of these block diagrams have several common features. First, both languages use a dataflow approach that comes naturally to engineers designing systems, embedded or otherwise. Second, the interface provides a high level of abstraction, thereby resulting in a clean interface that supports debugging. Third and most important for professors, both of these representations lend themselves naturally to teaching concepts using a step-by-step approach.

3. EMPOWERING DOMAIN EXPERTS

As noted in a previous section, the designers are not necessarily DSP-design experts. However, they are experts in their application area and want to use embedded platforms such as DSPs and FPGAs for their application because of the advantages that these platforms provide. Conventional programming methods increase the learning curve making it difficult to use DSPs or FPGAs for application development, especially for domain experts who are not embedded design engineers. Graphical programming languages alleviate this problem by providing a simple, easy-to-use interface that can be used to program DSPs, FPGAs and microcontrollers empowering the domain experts to quickly design, prototype and deploy systems. Figure 2 compares the steps that are required when using conventional and graphical programming techniques for DSPs. A similar chart can be drawn for other hardware platforms such as FPGAs or Microcontrollers.

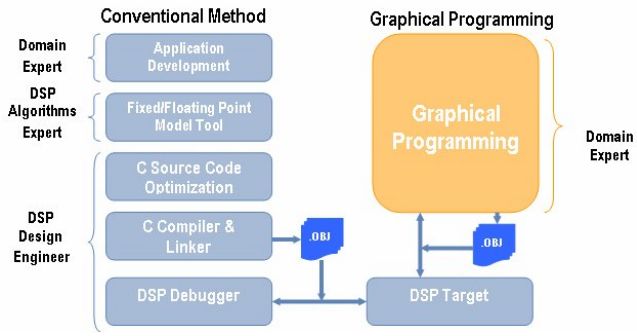


Figure 2. Abstracting complexities using graphical programming

The comparison between the two approaches in figure 2 present a very interesting perspective to teaching embedded systems to non-EE majors. The domain expert, in this case, the non-EE students are concerned with their design only. By providing a higher level of abstraction we are able to abstract the complexity in designing the embedded system helping the domain expert focus on the design at the same time teaching them how to use embedded systems.

4. A MARQUEE EXAMPLE - ChallengeX

ChallengeX[6] is a multi-year student competition in which seventeen teams have been challenged to re-engineer a GM Equinox, a crossover sport utility vehicle to minimize energy consumption, emissions, and greenhouse gases while maintaining or exceeding the vehicle's utility and performance.

The team that won ChallengeX in 2006, Virginia Polytechnic Institute and State University in the United States consisted primarily of mechanical engineers who are experts in the design of vehicles. In this competition, there was a need to use embedded systems that could be used to build complex Control Units and other regulatory parts for the Equinox. Figure 3 shows the overall architecture that Virginia Tech presented.

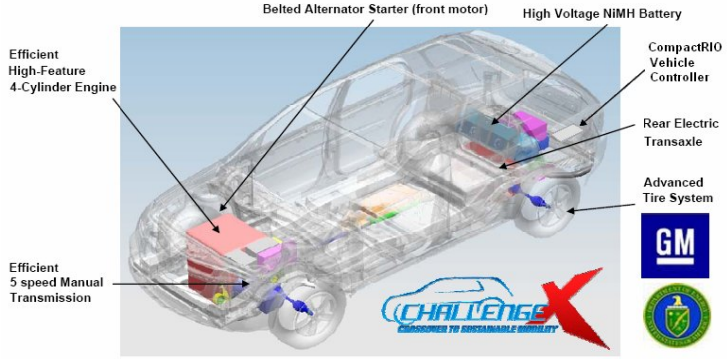


Figure 3. ChallengeX Vehicle design from Virginia Tech

Virginia Tech used graphical system design techniques to design their system. They used NI LabVIEW to program an FPGA-based platform, NI CompactRIO. The key point here is that the team was not comprised of embedded design experts, but because of they learnt embedded design using graphical techniques, they were able to design a highly efficient embedded control system.

5. ANOTHER EXAMPLE – LEGO

LEGO Mindstorms have been used by several professors and educators to demonstrate engineering concepts [7][8][9][10]. While it may seem as a toy, it is also a powerful robotics platform. For example, the new LEGO Mindstorms NXT brick is a 32-bit ARM processor that interfaces with sensors and motors and communicates with the host via Bluetooth. The key pedagogical element here is that the individuals for whom this "toy" is aimed at are typically between the 5-15 years of age group! The reason for this "toy" being successful is that it uses a completely graphical interface. Figure 4 shows the LEGO Mindstorms NXT software.

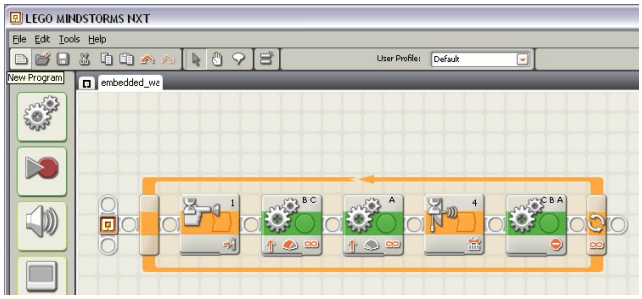


Figure 4. The next generation LEGO Mindstorms NXT software

Not only is this tool being used by high-school kids, but professors from engineering disciplines such as mechanical and aerospace systems are using this tool to help students learn about the nuances of embedded systems such as limitations on memory without having to worry about the implementation details such as memory allocation, pointers and netlist generation. The software takes care of all these details allowing the student to focus on the design of his or her system.

6. BENEFITS OF GRAPHICAL PROGRAMMING

6.1 Drag-and-drop interface

Graphical programming languages provide the user with a true drag-and-drop interface that reduces the learning curve drastically. Figure 5 shows an example of a block diagram from NI LabVIEW. Ptolemy has similar diagrams as well. From a pedagogical standpoint, the benefit for professors is that they can now teach the students using block diagrams using a chalk and a whiteboard and create the same block diagram in software that automatically translates and runs on hardware.

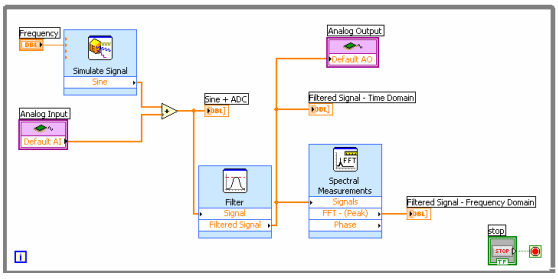


Figure 5. Drag-and-drop interface helps teach concepts easily

6.2 Hundreds of Pre-built Function Blocks

Designing embedded systems is incomplete without signal processing and control functions. As it was shown in figure 5, graphical programming environments aim to minimize textual code needed to design algorithms and provide users with a library of basic functions that can be used to develop embedded systems quickly. Graphical environments achieve this goal by providing pre-built function blocks that can be wired together to design a system. The advantage that graphical environments provide is that the same program that is used for simulation purposes can also be downloaded to supported targets and hence give the students access to real-world signals and enhance hands-on learning. Figure 6 shows some examples of time-domain function blocks.

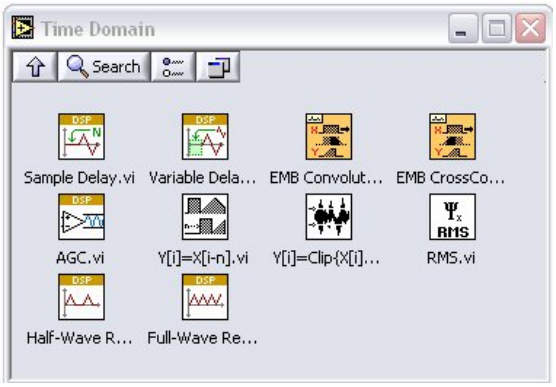


Figure 6. Some pre-built function blocks for time-domain

Professors can also teach students to build complex functions using the basic function blocks and hence promote the concept of reuse in embedded systems.

6.3 Ability to Re-use Existing C Code

Embedded design has traditionally involved a lot of C and assembly code. Professors and students may also have a lot of existing C code that they may wish to reuse. Graphical environments present a ways to reuse existing C code. The underlying implementation may vary for the different environments. One of the simplest techniques is shown in Figure 7. This technique is used in NI LabVIEW. Professors and students can insert their C code and create the input and output terminals and the compiler will compile any C code in the inline-C node into a header file and include it in the embedded project.

This also provides an opportunity for professors to introduce the finer aspects of embedded programming to non-EE majors. An example scenario may involve designing the application using the pre-built function blocks and then redesign the system by replacing the function blocks with C code that may involve custom optimization that enhance the performance of that particular application. In this way, students get to experience the intricacies of embedded programming while still being able to learn about designing the system by using the pre-built function blocks

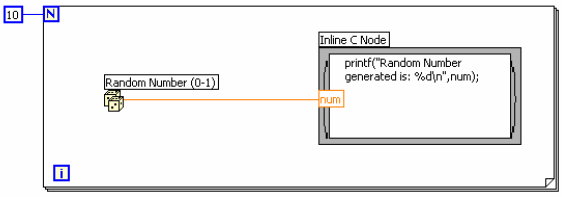


Figure 7. Reusing Existing C code in Graphical Environments

6.4 Inherent Concurrency and Parallelism

In text-based languages, implementation of parallelism often requires a complex balance of library calls to operating system functions, resource management, memory protection, and locking mechanisms. As a result, the compiler needs extensive code to be written to ensure shared sections of code are properly protected, making it cumbersome to build parallel programs. However, parallel programming forms the cornerstone for embedded architectures such as FPGAs that are used extensively in Academia. The same is true for several other applications that needs some form of parallelism. Because graphical programming is based on the dataflow approach, concurrency and parallelism are inherent making it easy to design such systems. Figure 8 shows an example of parallel programming where in input is acquired and filtered in the top loop and a triangle waveform is generated and output in the second loop.

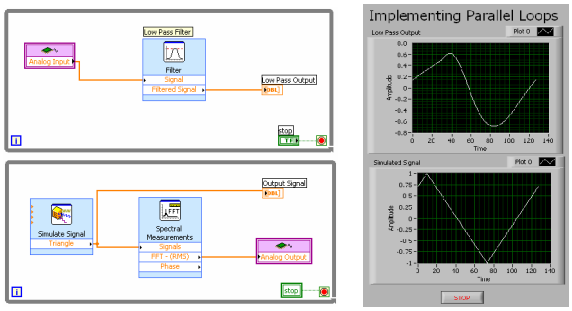


Figure 8. Implementing Parallelism in Graphical Programming

7. CONCLUSION

Embedded systems are pervasive and have moved away from being used only by embedded design experts. Because of the general adoption by "domain" experts in various areas, there is a need to teach the basics of embedded design to non-EE majors. Graphical programming helps address this challenge by providing a simple, easy-to-use interface that provides a platform with a higher level of abstraction allowing the domain experts to focus on their design and hence present a viable alternative for professors and students to learn to design embedded systems. In this paper, we presented an overview of graphical programming systems and explained their benefits. We have also outlined two use cases - ChallengeX and LEGO Mindstorms NXT – the next generation LEGO system that is targeted for the kindergarten and middle-school children that take advantage of the graphical programming approach.

8. REFERENCES

[1] A. Benveniste and P. Le Guernic, "Hybrid Dynamical Systems Theory and the SIGNAL Language," *IEEE Tr. on Automatic Control*, Vol. 35, No. 5, pp. 525-546, May 1990

[2] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud, "The Synchronous Data Flow Programming Language LUSTRE,"*Proceedings of the IEEE*, Vol. 79, No. 9, 1991, pp. 1305-1319

[3] P. Hilfinger, "A High-Level Language and Silicon Compiler for Digital Signal Processing", Proceedings of the *Custom Integrated Circuits Conference*, IEEE Computer Society Press, Los Alamitos, CA 1985, pp 213-216

[4] The Ptolemy Project http://ptolemy.eecs.berkeley.edu/

[5] Overview of LabVIEW
http://zone.ni.com/devzone/conceptd.nsf/webmain/F34045D2CC5357F486256D3400648C0F?OpenDocument&node=200067_us

[6] ChallengeX tournament official website

http://www.challengex.org/

[7] Teaching Engineering through LEGO mindstorms
http://www.areeonline.org/?id=5193

[8] LEGO Robotics in Engineering
www.asee.org/acPapers/00638_2001.pdf

[9] ROBOLAB @ CEEO website
http://www.ceeo.tufts.edu/robolabatceeo

[10] LEGO Mindstorms NXT System
http://mindstorms.lego.com/