DSP SYSTEM DESIGN USING LABVIEW AND SIMULINK: A COMPARATIVE EVALUATION

N. Kehtarnavaz and C. Gope

Department of Electrical Engineering, University of Texas at Dallas

ABSTRACT

LabVIEW and Simulink are two most widely used graphical programming environments for designing digital signal processing (DSP) systems. Unlike conventional text-based programming languages, such as C and MATLAB, graphical programming involves block-based code development, allowing a more efficient mechanism to build and analyze DSP systems. This paper presents a comparative evaluation between LabVIEW and Simulink in terms of a number of ease-of-use and functionality criteria. Twenty students taking a senior undergraduate DSP lab course were asked to perform the evaluation. The students' responses indicate that these two graphical environments provide more or less the same design features with LabVIEW having an edge over Simulink as far as graphical display/visualization and DSP hardware integration tools are concerned.

1. INTRODUCTION

LabVIEW and Simulink are two widely used graphical code development environments which allow system-level developers to perform rapid prototyping and testing. Unlike text-based programming languages, such as C, MATLAB, and Java, graphical programming involves block-based code development and offers a more intuitive approach to designing systems.

The comparative evaluation reported in this paper is limited to the design of digital signal processing (DSP) systems. Seven DSP system design problems were assigned to senior undergraduate students as part of their laboratory work in a DSP lab course entitled "DSP Design Project" at the University of Texas at Dallas. The students were asked to build and analyze the DSP systems in LabVIEW as well as Simulink and compare their respective merits and demerits based on a number of evaluation criteria mentioned later in the paper.

Section 2 gives a brief overview of the LabVIEW and Simulink graphical programming environments. Section 3 describes the DSP system design problems, while section 4 explains the criteria used for the comparative evaluation.

The results obtained appear in section 5, and the conclusions are stated in section 6.

2. GRAPHICAL PROGRAMMING: LABVIEW AND SIMULINK

A typical graphical code consists of various blocks interconnected by wires. The blocks (which might consist of other sub-blocks) are the processing units and the wires are responsible for transferring data from one block to another. Graphical programming is based on the concept of data flow, in contrast to the sequential logic of most text-based programming languages. This means that the execution of a block or a graphical component is dependent on the flow of data, or more specifically a block executes when data are made available at all of its inputs, and output data of the block are sent to all other connected blocks. Data flow programming allows multiple blocks to be run simultaneously since their executions are determined by the flow of data and not by sequential lines of code, which is the case in text-based programming.

The LabVIEW environment consists of two major components: Front Panel (FP) and Block Diagram (BD). An FP provides the graphical user interface while a BD contains the building blocks of a system and resembles a flowchart. LabVIEW systems are called Virtual Instruments (VIs) and its FP appears as an instrument panel consisting of various controls and displays. The interested reader is referred to [1] for details on the LabVIEW programming environment.

Similar to LabVIEW, Simulink offers a block-based programming approach for simulation, design, and analysis of dynamic systems. It provides an interactive graphical environment together with a set of libraries to design and simulate systems including DSP systems. Simulink blocks are called Models and unlike LabVIEW, the code implementation and input/output entities are not distinguished explicitly in Simulink. Simulink is integrated with MATLAB and hence can access the functionalities

and tools available within the MATLAB environment. The interested reader is referred to [2] for details on the Simulink programming environment.

3. DSP SYSTEM DESIGN LABS

This section briefly describes the DSP systems designed in the LabVIEW and Simulink environments for the evaluation reported later in the paper.

I. **Analog-to-Digital (A/D) signal conversion**: The goal of this lab was to study the sampling and quantization aspects of A/D signal conversion. In the first part, a discrete sinusoid signal was generated and sampled at various sampling frequencies to study aliasing effects. In the second part, a 3-bit A/D converter was used to represent an analog signal having values in the range 0-7. The system was required to display the quantization error and the associated error histogram.

II. **FIR/IIR filtering system design**: This lab dealt with the design and implementation of FIR/IIR filtering systems using the filter design tools available within the LabVIEW and Simulink environments. Given the filter specifications, LabVIEW's Digital Filter Design (DFD) toolkit and Simulink's Filter Design and Analysis tool (FDATool) were used to realize the filters. A low-pass FIR filter and a bandpass IIR filter were designed and used to filter some input signals, which were also generated within the environment.

III. **Fixed-point and floating point arithmetic**: The goal of this lab was to study digital filtering as implemented on fixed and floating point DSP processors, in particular on the TI TMS320C6000 DSP processor [3]. The Simulink Fixed Point toolset was used to simulate fixed-point arithmetic for the TI TMS320C6000 DSP. At the time, LabVIEW did not have any such capabilities and hence the fixed-point arithmetic was performed using the Q-format representation [3].

IV. **TI DSP integration**: LabVIEW and MATLAB, both offer tools to visualize, verify, and validate TI DSP code by integrating the graphical environment with the TI Code Composer Studio (CCS) development tool. Using the Real Time Data Exchange (RTDX) feature of the TMS320C6x DSP, data can be transferred to and from a DSK board connected to a PC and be analyzed in the LabVIEW or MATLAB environment in real-time. In this lab, the students were asked to use the MATLAB's *Link for CCS* tool and LabVIEW's *DSP Test Integration* tool to generate signals in the MATLAB and LabVIEW environments, respectively, and send signal samples to the C6416 and C6713 DSK boards connected to the PC, filter the signals on the DSP, and send the filtered signals back to the MATLAB or LabVIEW for display and analysis. Although graphical programming with LabVIEW allowed the DSP integration, Simulink did not offer any such capabilities and hence here the MATLAB text-based Link for the CCS tool was used.

V. **Adaptive filtering**: In this lab, the students were asked to build an adaptive FIR filtering system using the Least Mean Square (LMS) algorithm, to perform adaptive noise cancellation for a signal corrupted by time-varying noise. It should be noted that unlike LabVIEW, Simulink already had a built-in LMS block available with the Signal Processing blockset.

With the exception of the DSP integration lab, all the lab assignments were required to be implemented completely in software, including signal generation, processing, and display.

4. EVALUATION CRITERIA

Eight criteria were used to evaluate and compare the efficiency of the LabVIEW and Simulink graphical programming environments. For each design problem, the students were asked to rate the two environments on a scale of 0 to 10, with 10 representing the highest rating or score. A brief description of the evaluation criteria is provided below:

- **Learning curve**: This criterion reflected the duration of getting familiar with the programming environment. Most of the students had no or very little exposure to graphical programming environments and were expected to learn how to build systems in LabVIEW and Simulink.

- **Ease of use**: Assuming sufficient familiarity with the environment, this measure of evaluation reflected the ease of operating in the environment to develop and modify code. In other words, the students were asked to evaluate whether the environment offered easy to use features for code development, reuse, and expansion.

- **Programming constructs**: This criterion was used to evaluate if sufficient programming constructs and data structures were available in the graphical programming environment, for example, *while* and *for* loops, *if else* constructs, *switch case* constructs, ability to make function calls, hierarchical code organization, etc.

- **Breadth of functionality**: A major motivation behind developing DSP systems via block-based graphical programming is the ability to use off-the-shelf functional blocks or components to build complex systems. Thus, this criterion was used to evaluate the extent the programming environment offered a rich set of plug-and-play building blocks for DSP system design.

- **Graphical User Interface (GUI)**: One of the main attractions of graphical programming environments is its GUI. Once a system is designed and implemented, its GUI allows one to easily interact with the system to visualize and analyze its behavior. The students were asked to evaluate the GUI capabilities of the two programming environments.

- **Debugging features**: Code debugging usually consumes a significant portion of the code development process and efficient debugging tools are a key requirement for any programming environment. This criterion was used to evaluate the debugging features available in the environment, in particular the graphical debugging tools.

- **DSP Test Integration tools**: This criterion reflected whether it was easy to extend the environment to integrate it with other software and hardware platforms. Specifically, the students were asked to evaluate if it was easy to interact with the TI Code Composer Studio tools and hence integrate the graphical programming environment with the TI C6416 and C6713 DSK boards.

- **Help resources**: Graphical programming mostly involves block or component based programming, wherein functional building blocks are made available to system developers. For example, for DSP system development, tools such as Signal Generation, Filter Design, Fast Fourier Transform, Power Spectral Density Estimation, Waveform Measurements, etc., are readily available and it is essential to have rich technical documentations and if required, online help, be readily available to system developers. This criterion reflected the richness of the help resources offered by the environment.

5. COMPARISON RESULTS

This section presents the evaluation and comparison results as related to the design problems described in section 3 and the criteria mentioned in section 4. Twenty students were asked to do the evaluation and comparison. On an average, the students took 2 to 4 lab hours, depending upon the complexity of the design problem, to finish each lab exercise. Moreover, the criterion "Ease of use" reflects the ease of implementing a system design in LabVIEW and Simulink environments. Figure 1 shows the students' scores for each of the criteria, averaged first over all the design problems and then over all the students.

As can be seen from this figure, within the error margins, LabVIEW and Simulink were rated almost equally for most of the criteria. The most conspicuous disparity between the ratings occurred for the Graphical User Interface criterion. LabVIEW was preferred over Simulink by a wide margin for its easy-to-use GUI capabilities. This feature is an integral part of the LabVIEW VI structure for code development. The other two criteria for which LabVIEW was slightly preferred over Simulink included its easier to use TI CCS integration tools and richer help resources.

As an example, Figure 2 shows the graphical codes for the filtering system (Lab II) for LabVIEW (Block Diagram) and Simulink (Model) and Figure 3 shows the Front Panel associated with the LabVIEW Block Diagram. As displayed on this Front Panel, the parameters for the input signals can be changed on the fly and their output effects can be seen in real-time.

6. CONCLUSION

This paper has provided a comparative study between LabVIEW and Simulink, two popular graphical programming environments, in order to evaluate their effectiveness in building DSP systems for educational purposes. Based on the ratings of twenty students over eight criteria, it is concluded that both of these graphical programming environments can be used to shorten the amount of code development time, with LabVIEW having an edge with respect to GUI features and DSP integration tools. The authors hope that this comparative study was useful for the readers as well as the students who attended the course, by enabling them to make a more informed decision regarding the choice of programming environment for DSP system design.

REFERENCES

[1] R. Bishop, *Learning with LabVIEW 7 Express*, New Jersey: Prentice Hall, 2004.

[2] J. Dabney and T. Harman, *Mastering Simulink*, Prentice Hall, September 2003.

[3] N. Kehtarnavaz, *Real-Time Digital Signal Processing Based on the TMS320C6000*, Elsevier/Newnes, 2004.

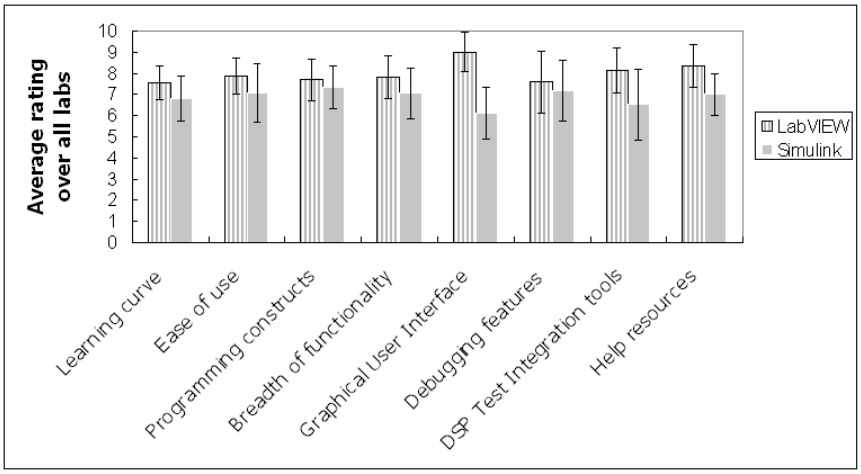

Fig.1 - Comparison of LabVIEW and Simulink graphical programming environments.

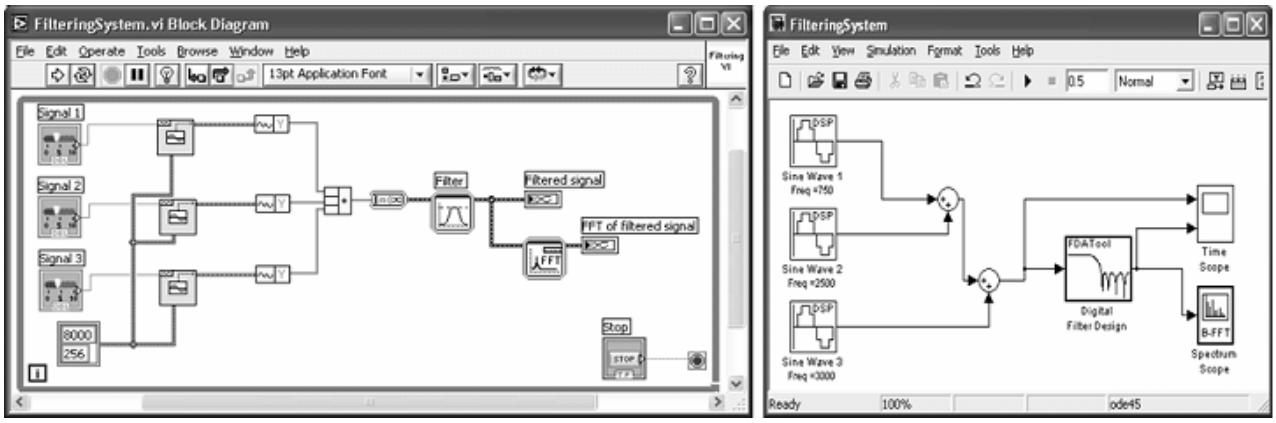

Fig. 2 - Filtering system graphical codes in LabVIEW (left) and Simulink (right).

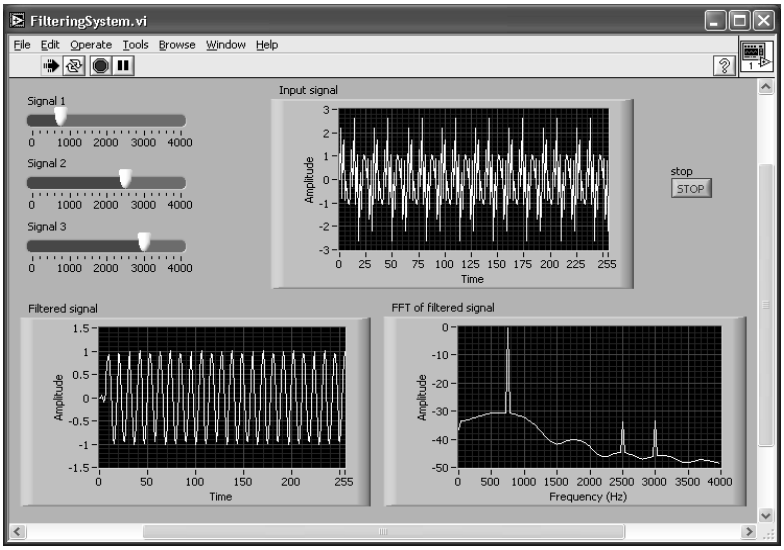

Fig. 3 - Filtering system *Front Panel* in LabVIEW.