AN INTEGRATED ENVIRONMENT FOR DEVELOPING REAL-TIME DSP APPLICATIONS

Jacob Fainguelernt
Tel-Aviv University
Tel-Aviv, Israel
jfaing@eng.tau.ac.il

Graham Reith
The MathWorks Ltd.
Cambridge, UK
Graham.Reith@mathworks.co.uk

Richard Sikora
JLR Engineering Centre
Coventry, UK
richard_sikora@hotmail.com

ABSTRACT

This paper describes an innovative environment to develop real-time embedded Signal Processing based applications, and a suite of example applications that have been developed to assist in academic teaching of practical signal processing. The objective of this environment is algorithmic design and simulation, which itself is educational but it can also be used as a rapid prototyping tool, enabling students to experiment with real-time DSP applications in a variety of fields. The environment offers a wide range of libraries to support this, and it has an open hardware and software architecture enabling further expansion of its functionality.

***Index Terms*—** Signal processing, Engineering education, Educational technology**,** Real-time systems, Programming environments

1. INTRODUCTION AND MOTIVATION

Significant work has been done within the field of DSP education, trying to bridge the gap between mathematical theory and real-time implementation. Educational labs and courses have been developed in the past, some of which are based on DSP programming [1][2] whilst others rely solely on the MATLAB environment. In this case MATLAB and Simulink are used, together with Real-time Workshop [3] code generation technology, allowing high-level algorithms to be developed and understood in simulation whilst also allowing C code to be developed and run on a COTS DSP board [4][5]. Recently code generation tools were introduced in other DSP courses [6],[7] making use of the Target for TI C2000^{TM} [8] and the Target for TI C6000^{TM} [9]. The above references however, address only the DSP student audience, as they only demonstrate algorithm implementation.

We present here a tool that can be used not only for education, but also as a prototyping tool for a wide range of DSP based applications. The tool can be used not only DSP specialists, but also by academic people wishing to implement an application on real-time hardware having a limited knowledge of DSP programming techniques.

2. OBJECTIVES

The main objective of this project was to create a suite of example applications that can be easily used by academic people belonging to either of the following groups:

- **Top-bottom designers:** Academics developing algorithms, systems and applications by first modeling and simulating in high-level languages like MATLAB M-code or Simulink block diagrams who want to prototype and deploy their ideas on TI DSPs, and who want to validate the design and its real-time characteristics on hardware.
- **Bottom-Top designers:** Practitioners very familiar with TI DSP hardware programming tools who want to tap into modeling and simulation capabilities of MATLAB and Simulink and bring their application to higher levels of abstraction, either in M-code or Simulink models to study design tradeoffs and test their algorithms in a system-level environment.

Students should be able to learn from the examples, and then create their own applications using the building blocks provided by the tool. They could then extend the system capability further by developing their own blocks in M or C and integrating them with the system model.

In order to cover this target audience, the following requirements were applied:

- **Intuitive User Interface** – Students should be able to have a quick and easy interaction with the environment both in the development phase and when interacting with the implemented design.
- **Open Architecture –** Well-defined hardware and software interfaces should allow external modules (hardware and software), to be easily integrated within the environment as new building blocks.
- **Smooth Migration from Simulation to Real-Time –** The simulation environment should reflect real-world environmental effects with maximum accuracy.
- **Target Platform Portability –** Algorithms should be easily ported from one COTS DSP board to another, as far as system interfaces and device performances allow.

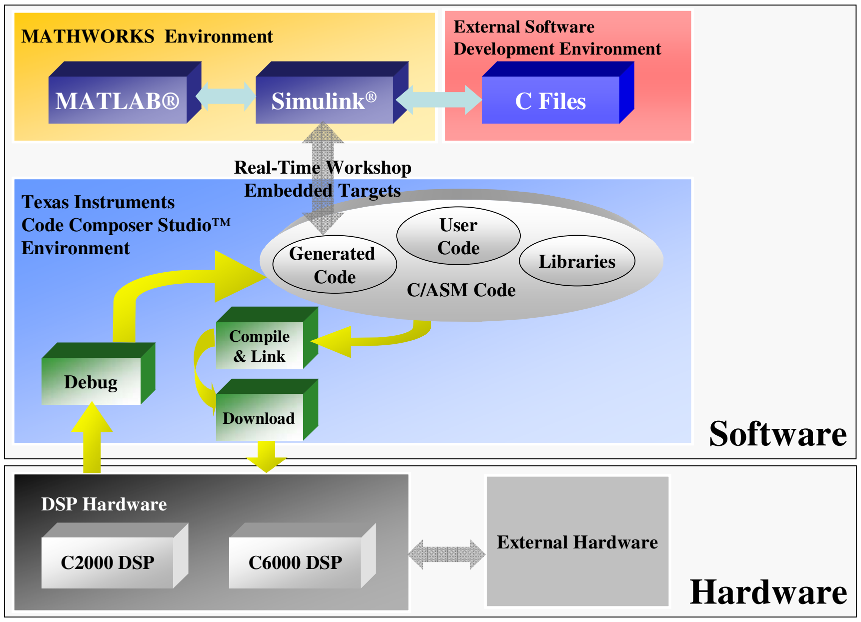

Figure 1 - The Development Environment

4. THE ENVIRONMENT

The development process for a typical DSP application comprises three stages. It starts with a top level design, where algorithms are designed and simulated. From this top-level, C code is generated which is passed to the compiler environment, together with required device drivers and a scheduler. This project is then compiled and downloaded to the DSP, from where debug links can return verification data back to the top-level environment.

Figure 1 shows the building blocks of the development environment. It is based on Simulink for the top-level design and C code generation using the ***Real-Time Workshop, Real-Time Workshop Embedded Coder, Link for Code Composer Studio, Target for TI C6000*** and/or ***Target for TI C2000*** tools. The environment supports a range of the TI DSP development boards. The ones used by the examples are outlined in Table 1.

Table 1- Hardware Platforms

Board	Application Field
eZDSPF2808 [10]	Control
eZDSP-F2812 [11]	Control
DSK6713 [12]	Audio and Communication
DSK6416 [13]	Imaging
DM6437 DVDP [14]	Video

The functionality of the basic configuration above can be expanded with additional block libraries for particular applications. For algorithm design purposes, MATLAB M-code and also C code can be used to create new Simulink blocks. The user can also integrate the code that is generated with code developed by himself and/or software libraries in the CCS environment. Standard analog and digital interfaces allow the DSP board to be connected to standard input/output devices (microphones, speakers, video cameras, motor drivers, etc …), external devices, and is extensible to allow inclusion of dedicated boards designed by the user.

5. METHODOLOGY

The use of the system is a four steps process as follows:

1. Create a Simulation Model in Simulink, using virtual sinks and sources (Simulink blocks).
2. Run the simulation on a PC and tune its parameters.
3. Replace simulation sources and sinks with real hardware interfaces. Generate the DSP code.
4. Run the system in real-time.

This process is illustrated by the example shown in Figure 2. In this example, the environment is used to build a physical model of vehicle engine management in simulation, which is then used to create an implementation on the eZDSP-F2812. This board will emulate in real-time the behavior of a vehicle, under various stimuli. The model contains two blocks representing the engine management and the vehicle dynamics (based on equations of motion). To simulate the gas pedal of a car being smoothly depressed, in the simulation, a ramp stimulus with saturation is applied and the vehicle behavior is verified using various process blocks placed along the model.

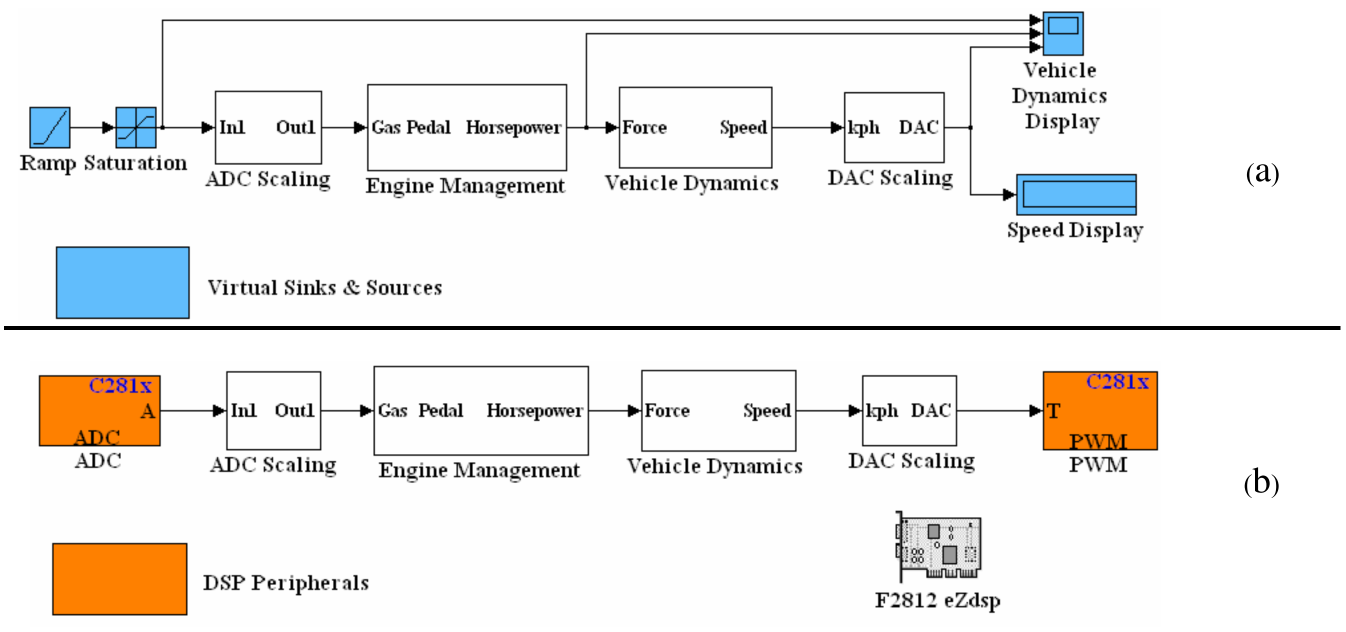

Figure 2 Vehicle Engine Management: Simulation Model (a) and Implementation Model (b)

In the second phase, the ramp stimulus is replaced by the Analog to Digital Converter Block of the TMS320F2812 and the speed display is replaced by a PWM generator block. This enables the use of a signal generator to stimulate the physical model and analyze the behavior with an oscilloscope connected to the PWM output.

6. ADVANCED FUNCTIONALITY

The Car model described in the previous section exemplifies the basic use of the system. In the current section we describe (in brief) some other applications, which exemplify the use of a variety of system features for various fields of application.

These represent a small selection of the examples that have been developed as workshop and lecture material for educational establishments.

DC Motor Control: The goal of this example is to implement a real-time DC motor speed control system on the eZDSP-F2812.

The first step is to identify the DC motor transfer function. The DC motor was activated in open loop, its step response was analyzed using MATLAB tools, and the transfer function was estimated.

The DC motor control loop was then designed around this using a PID controller. The simulation model made use of a PID block is based on a library function optimized for the TMS320F2812. This block was used both for simulation and for real-time, allowing an accurate fixed-point simulation of the application and also optimal on-target performance.

An analog interface card was designed for this example based on the simulation results, and the electrical specifications of the eZDSP-F2812 and the DC motor.

Audio Conference Bridge: The Audio Conference Bridge example demonstrates the control of a voice call with multiple (n >2) attendants. The algorithm monitors the voice signals from all attendants, and switches the signals to be transmitted back to the attendants, according to a controlling algorithm,

This application cannot be implemented using solely the DSK6713, as it has only 2 analog ports. Therefore two daughterboards, the TLV320AIC24/24KEVM [15] and the DSP-CODEC development platform [16], are used.

Drivers for those boards are not available as part of the Target for TI C6000^{TM}, so a new Simulink driver block was created based on a legacy driver (developed in the Code Composer Studio (CCS) environment).

Lane Detection: This example illustrates an application for detecting the lane markings on a road as seen from a vehicle. This would form the basis of automotive driver assistance systems, eg Lane Departure Warning. The algorithm is based on detecting the two longest lines in the image and displaying a polygon based on those lines to indicate the current lane. First the region of interest (ROI) is determined, and this is processed by an edge detection algorithm. The two longest lines are then located using the Hough transform. The outputs of this are validated to improve the lane tracking performance, ignoring large variations in the Hough transform output values. The final block constructs the lane polygon superposed over the original image.

The lane detection model uses mainly standard Simulink blocks. An exception to this is the lane tracking performance improvement block, which is a logical function and can be more easily described and implemented by some MATLAB M-code.

Other Examples: Besides the applications above, a number of other applications have been developed as listed below:

Audio and Communication:	• Acoustic Noise Cancellation
	• Audio Echo & Reverberation
	• Wavelet Denoising
	• SNR Measurements
	• DTMF
	• Spectrum Estimation
	• Analog Modulation
Imaging:	• Edge Detection
	• Line and Lane Detection
	• Video Surveillance
Control:	• DC Motor Control
	• PMSM Control

7. CONCLUSIONS

This paper presented a tool for the implementation of real-time DSP applications, and introduced a suite of example applications developed at the School of Electrical Engineering of the Tel-Aviv University, and are now available as workshops and lectures for academic use [17].

The use of the Simulink code generation tools and COTS DSP boards allows a seamless prototyping and implementation process. The tool capabilities, and scope of the examples, can be further enhanced due to its open software and hardware architecture.

8. ACKNOWLEDGEMENTS

The authors would like to acknowledge the support from the Texas Instruments University program through Cathy Wicks and Robert Owen. They would like also thank The MathWorks Inc., especially to Amnon Gai, Houman Zarrinkoub, Maureen Maher and Ken Karnofsky for their support.

The authors are thankful to Ilan Fono (Tel-Aviv University - School of Mechanical Engineering), for the development of the DC motor control model, and to Elad Sity and Ori Altman (Tel-Aviv University - School of Electrical Engineering), for the development of the Audio Conference Bridge model.

11. REFERENCES

[1] W. S. Gan , "Teaching and Learning the Hows and Whys of Real-Time Digital Signal Processing", IEEE Transactions on Education, Vol. 45, no. 4, pp. 336-344, November 2002.

[2] Jacob Fainguelernt and Arie Yeredor, "Bridging the gap between DSP theory and real-time implementation," Proceedings of the European DSP Education and Research Symposium (EDERS2004), Birmingham, UK , 2004

[3] The MathWorks Inc., "Real-Time Workshop® User's Guide", Version 7.0, September 2007

[4] K. H. Hong and W. S. Gan, "Efficient block diagram synthesis of DSP kernels for the TMS320C6701," presented at the 2000 Int. Conf. Signal Process. Appl. Technol., Dallas, TX, October 2000.

[5] W. S. Gan, , Y. K. Chong, W Gong, and W. T. Tan, " Rapid Prototyping System for Teaching Real-Time Digital Signal Processing", IEEE Transactions on Education, Vol. 43, no. 1, pp. 19-24, February 2000.

[6] W. S. Gan and S. M. Kuo, "Transition from SIMULINK to MATLAB in Real-Time Digital Signal Processing Education", International Journal of Engineering Education, Vol. 21; No 4, pp. 587-595, 2005.

[7] S. Gannot and V. Avrin, "A SIMULINK and Texas Instruments C6713 based Digital Signal Processing Laboratory", Proceedings of the 2006 European Signal Processing Conference (EUSIPCO), Florence, Italy, September 2006.

[8] The MathWorks Inc., "Target for the TI C2000™ User's Guide", Version 2.3, September 2007**.**

[9] The MathWorks Inc., " Target for the TI C6000™ User's Guide", Version 3.3, September 2007**.**

[10] Spectrum Digital Incorporated, "eZdspF2808 Technical Reference", Rev. C, October 2005.

[11] Spectrum Digital Incorporated, "eZdspF2812 Technical Reference", Rev. F, September 2003.

[12] Spectrum Digital Incorporated, "DSK6713 Technical Reference", Rev. B, November 2003

[13] Spectrum Digital Incorporated, "DSK6416T Technical Reference", Rev. A, November 2004

[14] Texas Instruments: DM6437 Digital Video Development Platform,

[15] Texas Instruments, "TLV320AIC20K/24KEVM User's Guide", SLAU088A, April 2005.

[16] Texas Instruments, "DSP–CODEC Development Platform User's Guide", SLAU090, September 2002

[17] J. Fainguelernt, G. Reith and R. Sikora, " From MatLab to Real-Time with TI DSP", Texas Instruments & The MathWorks Inc document **DSP14105U**, To be Released December 2007. Available from www.ti.com/university & www.mathworks.com