Incorporating System-Level Design Tools into Upper-Level Electrical Engineering Courses

Wagdy H Mahmoud
University of the District of Columbia
wmahmoud@udc.edu

Abstract

This paper provides details of our electrical engineering efforts to vertically integrate system-level design tools with many junior, upper-level, and capstone courses. The paper also provides the details of developed laboratory exercises and projects in the areas of image processing, digital communication, and embedded processing. In addition, the paper details the challenges involved in using continually-evolving, system-level design tools and the efforts made to reduce their learning times.

Introduction

The Accreditation Board for Engineering and Technology (ABET) requires providing students with a significant hands-on interdisciplinary design experience. Graduating electrical engineering students should have the ability to design systems, subsystems, and components for real-time, application-specific hardware systems. They should also be able to implement these designs in functional hardware and test the hardware in real-world operating conditions. Students need to understand that their choices of implementation algorithm, style of functional units' implementations (serial, parallel, or pipelines), variable widths, and implementation board have a major impact on the quality of their designs.

Contemporary Electronic Design Automation (EDA) industry projects exhibit the following trends: a) increased design complexities; b) shorter time-to-market; c) increased dependency on intellectual property (IP) cores; d) hardware/software co-design; and e) system on a chip (SoC).

To improve graduating engineering students' productivities and prepare them for professional careers in the EDA industry, system-level design tools are being introduced in the upper-level and capstone design courses of the electrical engineering curriculum. To achieve such professional competence, students should be required to participate in a sequence of system-level design experiments and projects. These laboratory exercises aim at: a) familiarizing students with available design tools and implementation FPGA boards; b) sharpening students' abilities to design complex systems that include software and hardware components and to interface these designs to peripheral devices; c) exposing students to contemporary soft-core and hard-core processors; d) preparing students to design and implement useful applications; and e) providing students with lessons that can help them become life-long learners and successful professionals. The experiments provide the framework that guide students through a top-down approach to defining the problem;

devising complete design specifications; partitioning the design into cooperating hardware and software functional blocks; generating alternative forms of design for each block; and comparing the costs, area, power consumption, development time, and performances of these alternatives when implemented on real hardware (FPGA). The interdisciplinary design projects emphasize the hardware/software co-design and implementation of applications, the design and use of intellectual property (IP) core libraries, and system-on-a-chip (SoC) concept.

The system-level software packages used include MATLAB, Simulink, assortment of MATLAB toolboxes and blocksets, Xilinx Integrated System Environment (ISE), Xilinx System Generator (SysGen), Xilinx embedded design kit (EDK), Xilinx ChipScope, Xilinx AccelDSP, and ModelSim from Mentor Graphics. MATLAB toolboxes and blocksets provide commonly used design components and parameterized functions needed to model the majority of digital control, DSP, and digital communication systems. For maximum accuracy, MATLAB uses a double precision format to represent data. The SysGen provides a system-level design environment for FPGAs. SysGen is a toolbox that runs within the Simulink environment to produce HDL synthesizable code. SysGen also allows an HDL co-simulation and hardware-in-the-loop to verify the correctness of the produced design. The SysGen provides the resources to quantize double-precision data into a fixed-point format, which is suitable for the hardware implementation. The resource estimation utility of the SysGen provides an estimation of the hardware resources needed for the hardware implementation of the design using the target FPGA chip. The EDK allows the user to incorporate soft-core processors (Microblaze™ or PicoBlaze™) or to interface the built-in PowerPC processors with the reconfigurable FPGA resources. Such processors can also be used to interface the hardware system to a variety of input/output peripheral devices needed to supply input data to the hardware system and/or to display the results of processing. Both SysGen and EDK allow interfaces to IP core libraries. The use of IP cores allow designers to concentrate on the overall system design and performance without having to spend time to verify the correctness or performance of the system's components. The ModelSim simulation package provides both functional and timing simulation for the hardware design. The use of system-level tools provides students with the technical skills needed for an integrative approach to real-time digital systems design, simulation, testing, and implementation. Students also acquire significant hands-on experience in design, development, and verification of complex digital systems that have potential societal benefits.

The FPGA boards include the Spartan-3, Spartan-3E Starter Kit, Spartan-3A DSP, the XUP-V2Pro, and the DO-ML403-EDKISE- PC4-US. The reconfigurable resources of the Spartan-3E FPGA chip can be used to implement PicoBlaze™ and Microblaze™ soft-core processors. The V2Pro board contains one V2pro FPGA chip with two IBM PowerPC embedded processors, Audio Codec, and connectors to gigabit serial input/output. The DO-ML403 board contains a Virtex-4 FX chip that is optimized for embedded processing and supports both the PowerPC™ hard and MicroBlaze™ soft processors. These boards allow designers to interface with pre-designed IP cores or with custom-built IP cores. They can be used to implement a configurable system-on-a-chip (CSoc) for DSP and embedded applications.

The following sections of this paper provide a brief overview of some of the tools used and present the details of some of the laboratory exercises and projects in the areas of embedded processing, digital signal processing, and communication systems.

Embedded Processing Tools and Projects

According to the 2006 annual embedded market survey, embedded processors account for more than 98 percent of the processor market, and most new embedded system designs are using 32-bit processors. Unlike general-purpose processors that are optimized for a mix of I/O and compute-intensive applications, embedded processors are low-cost processors that are optimized to perform a specific function. Embedded processors are extensively used in a wide variety of applications, such as mobile phone systems, automotive applications, office and home equipments, aerospace applications, and defense systems. Therefore, embedded system design courses are used to teach interdisciplinary designs.

The embedded design interdisciplinary capstone design projects presented in this paper emphasize the use of hardware/software co-design methodologies, the design and use of intellectual property (IP) soft-cores such as the PicoBlaze and the MicroBlaze, hardwired embedded processors such as the PowerPC 405 and system-on-a-chip (SoC) concept, and high-level languages programs for designing embedded controllers and applications.

The PicoBlaze Architecture

The PicoBlaze is an 8-bit RISC microcontroller soft-core that can be efficiently embedded on the fabric of a variety of Xilinx Field Programmable Gate Arrays (FPGA) chips. The PicoBlaze microcontroller is a synthesizable VHDL source code that consumes 96 slice cells when implemented on a Spartan-3 XC3200 chip (approximately 5 percent of the chip total logic capacity). The microcontroller has 16 byte-wide general-purpose registers, a byte-wide ALU, 64 byte-wide internal general-purpose scratchpad RAM, CALL/RETURN stack up to 256 input and up to 256 output ports [1]. It can execute approximately 1K sequential instructions stored in its memory within the FPGA chip, typically in a single 1Kx18-block RAM. The PicoBlaze has three versions of the assembler; each is optimized for a specific set of Xilinx FPGA chips. In this effort, the Constant (K) Coded Programmable State Machine (KCPSM3) assembler is utilized. The PicoBlaze instruction set is close to the 8086/8088 instruction set. The complete instruction set for the PicoBlaze can be found in reference [1]. The microcontroller executes each instruction in two clock cycles and can deliver approximately 44 MIPS in a Spartan-3 FPGA and approximately 100 MIPS in a Virtex-4 FPGA. The PicoBlaze microcontroller is designed for non-timing control functions and managing peripheral operations.

PicoBlaze Laboratory Exercises and Projects

The following set of PicoBlaze laboratory exercises and tutorials are used to teach students the details of the PicoBlaze soft-core microcontroller and how it can be used to design an embedded application

a. Implement 8-bit by 8-bit, multiply and divide functions using bit-wise and, add and shift instructions available in PicoBlaze KCPSM3 assembly.

b. Use the scratchpad RAM to display data on the 7-segment LEDs available on the Spartan-3 board or LCD displays available on the Spartan-3E board.
c. FPGA implementation of an embedded processor with multiple peripherals, each of which has a different level of priority. This exercise introduces the concept of PicoBlaze interrupts.
d. PicoBlaze microcontroller implementation within the Xilinx System Generator environment.
e. Implement a simple encryption algorithm or a pulse width modulation function using the PicoBlaze KCPSM3 assembly.

The MicroBlaze™ Architecture

The MicroBlaze is a 32-bit RISC Harvard soft processor core that can be embedded in the reconfigurable logic of an FPGA chip [2]. The MicroBlaze has (32) 32-bit general-purpose registers, up to 14 special-purpose registers, arithmetic and logic unit (ALU), 32-bit data bus, 32-bit address bus, 32-bit instruction words, two levels of interrupts, and a single issue pipeline that can be configured as three stages or five stages. The MicroBlaze core is parameterized to allow enabling of a set of configurable features. The MicroBlaze has an extensive instruction set, with many of the operations having large number of variants for register, immediate, constant, signed, and unsigned operands [2]. This basic design can be configured with advanced features such as barrel shifter, memory management/memory protection unit, floating-point unit (FPU), caches, exception handling, and debug logic. In addition, some of these instructions, such as floating point multiplication and division, are optional in various hardware configurations. Optional support is provided for privilege mode, memory protection, and virtual address translation. The maximum performance of the MicroBlaze processor depends on the targeted FPGA chip and on the processor configuration. The fast simplex link (FSL) of the MicroBlaze allows users to interface an embedded or external hardware co-processor to accelerate the execution of compute-intensive, time-critical functions. The MicroBlaze soft-core processor is a major component of the Xilinx Embedded Development Kit (EDK). The EDK also includes the Xilinx Platform Studio™ Tool Suite (XPS) and a library of peripheral IP soft-cores. More information about the EDK concepts, tools, and techniques are found in reference [3].

MicroBlaze Laboratory Exercises and Projects

The following set of MicroBlaze laboratory exercises aims at familiarizing students with the EDK and MicroBlaze. These exercises target the Spartan-3E and the Spartan-3AN boards. The list of exercises includes:

a. Creating a simple hardware and software processor system using the Base System Builder (BSB) and the Xilinx Intellectual Properties (IPs) available in the EDK. This exercise familiarizes students with the EDK environment and how to use it to design a reconfigurable embedded processing system, as well as how to connect it to peripheral devices. The I/O interface uses a universal asynchronous receiver/transmitter (UART) for serial communication. The UART is designed using an IP core. Other peripheral devices are also created using an available IP core library. The MicroBlaze generates the implementation netlists of the processor and the FPGA implementation bit stream.

Downloading the implementation bit stream onto the development FPGA board, the processor system is realized.

b. Designing an information display system to display user-defined messages and information using a hyper-terminal.
c. Designing a MicroBlaze embedded processor with multiple peripherals using both vendor-supplied and user-created custom IP cores, each of which has a different level of priority. This exercise enables students to work with interrupt in the MicroBlaze environment. Implement the design in the FPGA board.
d. Designing a controller for an external flash memory. The flash memory can be used to store the processor program that is used to configure the FPGA chip automatically whenever the power is turned on.

For each designed system, the HW/SW system debug using the Xilinx ChipScope tool is performed.

The PowerPC 405 Processor Architecture

The PowerPC 405 is a 32-bit embedded RISC processor core that is designed for SoC applications [4]. Multiple PowerPC 405 cores are embedded inside the Virtex-II Pro and inside the Virtex-4 FPGA devices. The PowerPC 405 processor is a Harvard architecture with (32) 32-bit general-purpose registers, five-stage data path pipeline, ALU with multiply and divide units, auxiliary processing unit (APU), 16Kb data and instruction caches, IBM CoreConnect bus architecture, embedded memory management unit, dedicated on-chip memory interface to block RAMs (BRAMs), and debug and trace support. The PowerPC 405 is a low-power consumption processor with operating frequency at approximately 450 MHz in the Vitex-4 FX and 400 MHz in the Virtex-II Pro devices.

PowerPC 405 Laboratory Exercises and Projects

The PowerPC 405 (PPC) laboratory exercises are similar to those of the MicroBlaze. The following set of PowerPC exercises aims at familiarizing students with the PowerPC 405 architecture and its instruction set. They also help students gain experience in designing useful hardware/software embedded applications that target the Virtex-II Pro or the Virtex-4 FX devices. The set of exercises include:

a. Creating a simple PPC processor to display user-defined messages and information using a hyperterminal. A UART is used to send displayed data to the hyperterminal.
b. Adding multiple peripherals to the PPC processor system using both vendor-supplied and user-created custom IP cores, each of which has a different level of priority. This exercise enables students to work with the PPC interrupt. The generated netlist for the design is used to implement the design in the FPGA chip.
c. Writing a basic software application to access the IP peripheral.
d. Adding a timer to the designed system.
e. Writing a software application that uses the timer.
f. Performing HW/SW system debug using the Xilinx ChipScope debugging tool.
g. Creating a core using Xilinx System Generator and adding it to a PPC 405 using the on-chip peripheral bus (OPB) and the processor local bus (PLB) interfaces.

After completing the above exercises, students are able to implement more complex projects similar to those implemented using the MicroBlaze. The project list includes:

a. Using the Virtex-II Pro board equipped with VDEC1 video decoder to design a video-decoding system. The system converts analog video signal into digital values and displays the output on a standard VGA display connected to the board's SVGA port. The PPC 405 is used for the signal conversion computations, and the MicroBlaze soft processor is used to control the entire system and its peripherals.
b. Using the V2Pro board equipped with Audio Codec AC97, connectors to gigabit serial input/output, a standard VGA display connected to the board's SVGA port, audio source, and a speaker to design audio processing system. The system captures and filters audio signals and plays back the filtered signal to the user's speakers. The filtered signals are also converted from the time domain to the frequency domain using a 64-point FFT. The sampled output of the FFT is displayed on the VGA monitor.

By implementing the same project in the MicroBlaze environment and the hardcore PPC, students gain valuable experience in both hardware and software implementations. Students evaluate and compare the designed projects in terms of design effort, performance, resource utilization, and cost.

Signal Processing Design Exercises

To prepare students for digital signal processing and communication systems design projects, students need to be familiarized with the SysGen tool. A set of laboratory exercises aimed at designing a digital FIR filter are used to teach students the capabilities of the SysGen design tool. The FIR design set of exercises includes:

a. Use the filter design toolbox to compute the FIR filter coefficients.
b. Use the SysGen to create a multiply-accumulate (MAC) unit, which is the basic building block of the FIR filter. This exercise familiarizes students with the process of building a subsystem using the SysGen and Simulink blocks.
c. Use the SysGen to create a parameterized dual-port RAM to store the FIR coefficients and data.
d. Create the necessary control logic, such as address generator, for the dual-port RAM.
e. Construct the FIR filter using the created MAC unit and the parameterized dual-port RAM, and test the design using Simulink sources. Verify the results using a spectrum scope.
f. Compile the design for the target FPGA chip and perform a hardware-in-the-loop verification. Figure 1 shows the hardware-in-the-loop simulation of the FIR filter.
g. Generate the VHDL code and implement the filter in the FPGA board using the ISE tools.

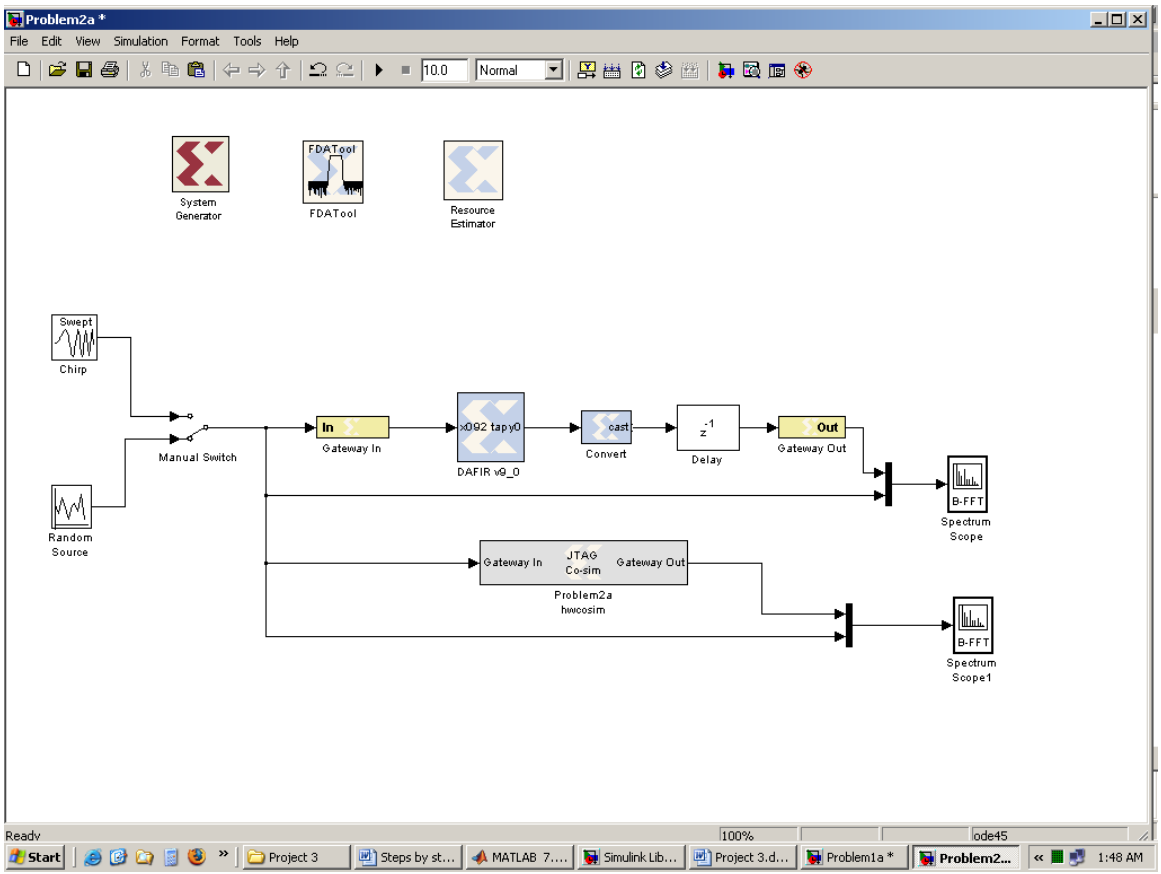

Figure 1: Hardware-in-the loop Simulation of the FIR Filter

By completing the above exercises, students gain substantial knowledge that enables them to implement more complex projects, such as a complex encryption algorithm, edge detection and feature extraction algorithms of images, or designing a controller for an industrial application. A description of the edge detection project follows.

The Edge Detection Project

Edge detection algorithms are basic and essential tools used in a variety of digital signal processing applications, such as feature extraction, target tracking, video surveillance, improving the appearance of blurred video streams, automated inspection of semiconductor chips finding the boundaries of organs in medical images, etc. MATLAB provides support for many methods for detecting edge in images such as Sobel, Prewitt, Roberts, Laplacian of Gaussian (LOG), Canny, and Zero-cross methods. Evaluating the performances of their algorithms and comparing their performances on images is beyond the scope of this paper. However, analysis of various edge-detection algorithms and their operators, the sequence of operation required for their use, and the choice of parameters can be found in references [5] and [6].

The first stage of this project involved using MATLAB coding and Simulink blocks to find the edge of an image. Figure 2 shows an original image, and Figures 3–7 details the results of these methods.

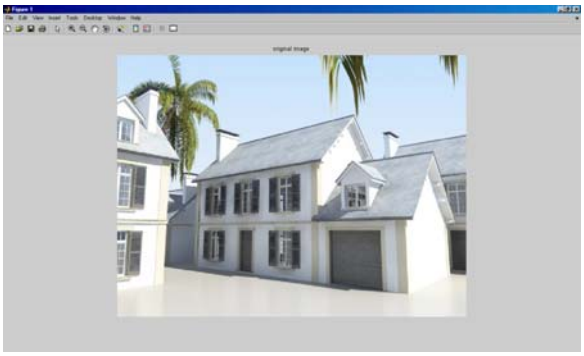

Figure 2: Original Image

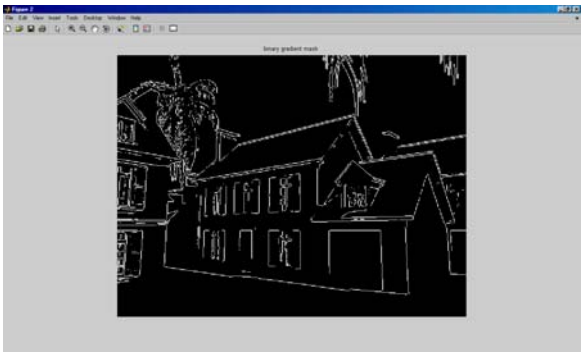

Figure 3: Edges using Sobel Method

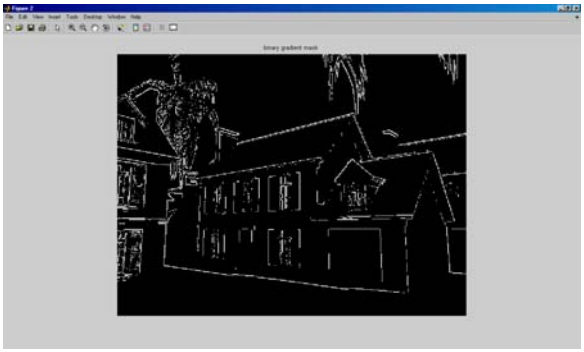

Figure 4: Edges Using Robert Method

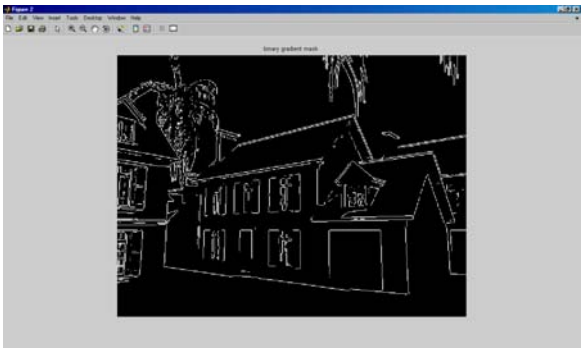

Figure 5: Edges Using Perwitt Method

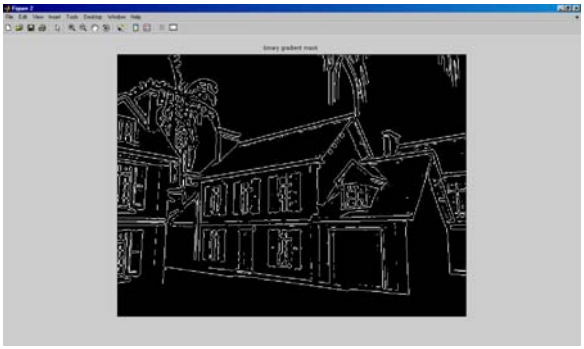

Figure 6: Edges Using LOG Method

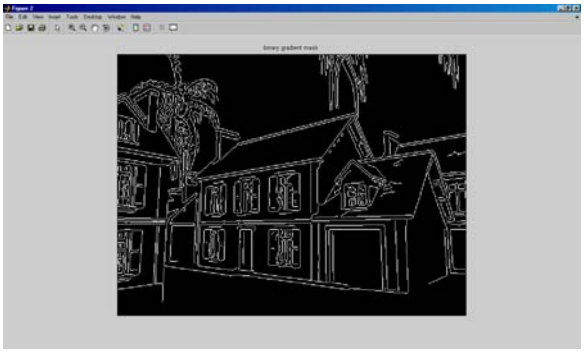

Figure 7: Edges Using Canny Method

The simulation blocks for implementing Sobel algorithms are shown in Figure 8. The Simulink blocks for implementing Canny, Sobel, and Prewitt methods are shown in Figure 9.

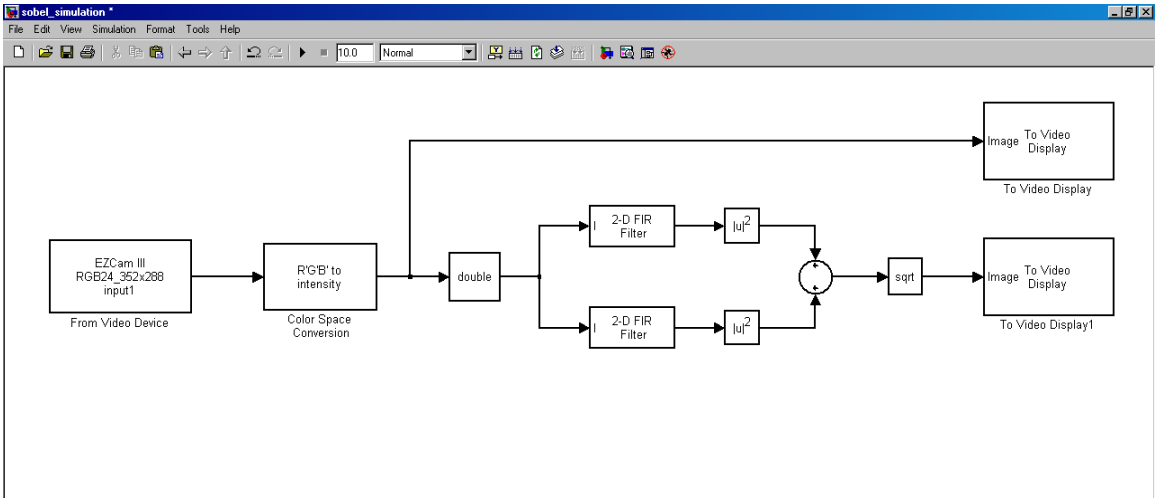

Figure 8: Simulink Blocks for Sobel Edge Detection

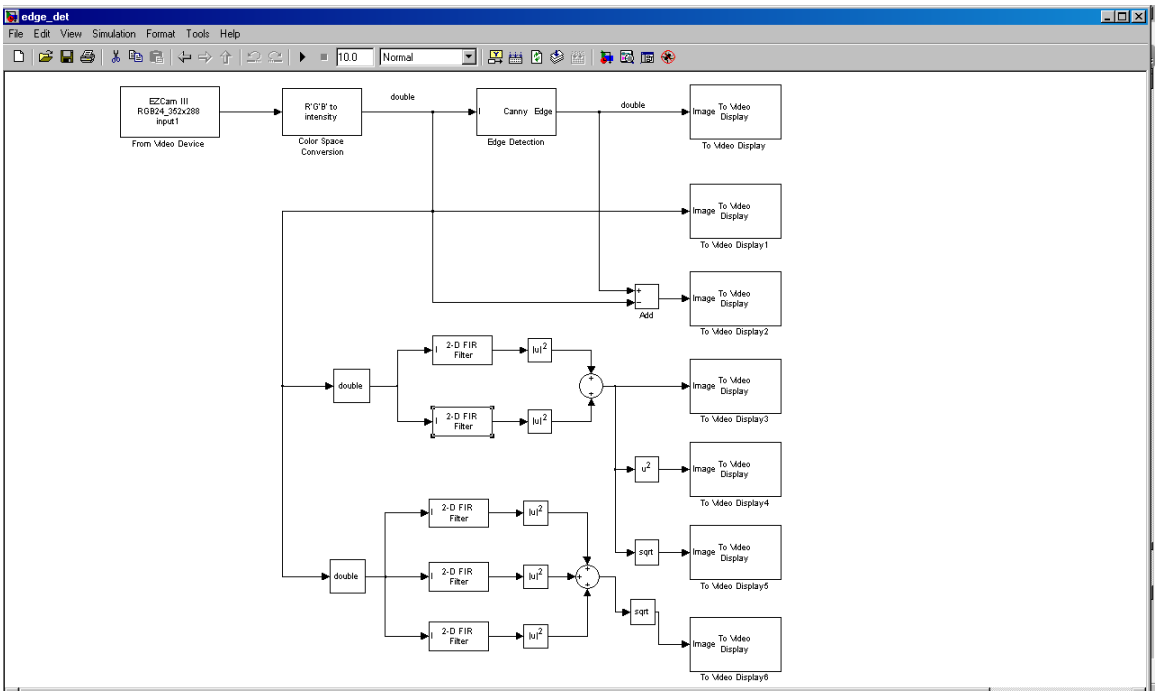

Figure 9: Simulink Implementation of Canny, Sobel, and Prewitt Methods

For real-time operation, Figure 10 shows the dialog block parameter needed to obtain live stream video from a Webcam.

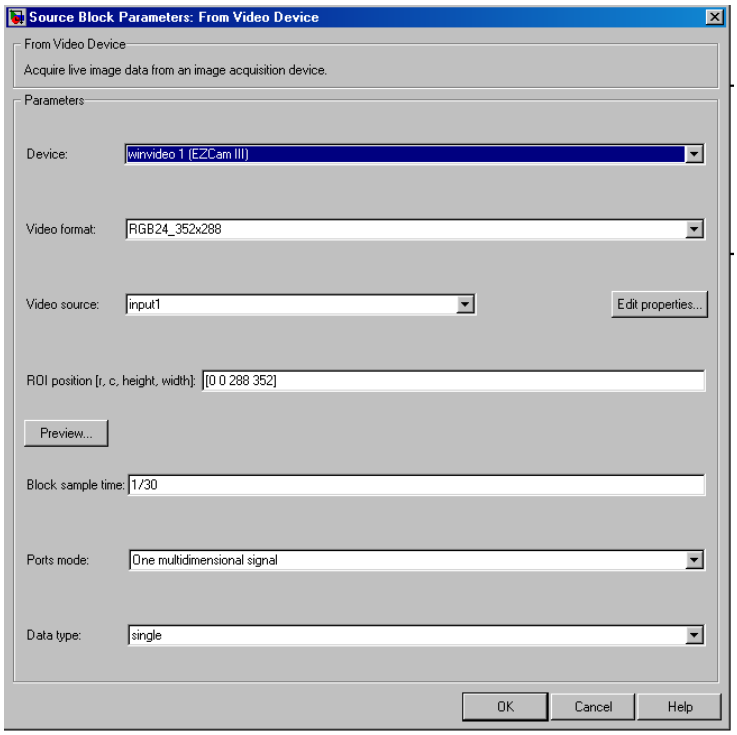

Figure 10: Block Parameter to Obtain Live Stream Video from Webcam

The user needs to convert the live stream video from RGB images to gray-scale images. The image noise needs to be removed using a 2-D filter. Figure 11 shows the simulation result of processing one frame of the video stream using five different edge detection algorithms.

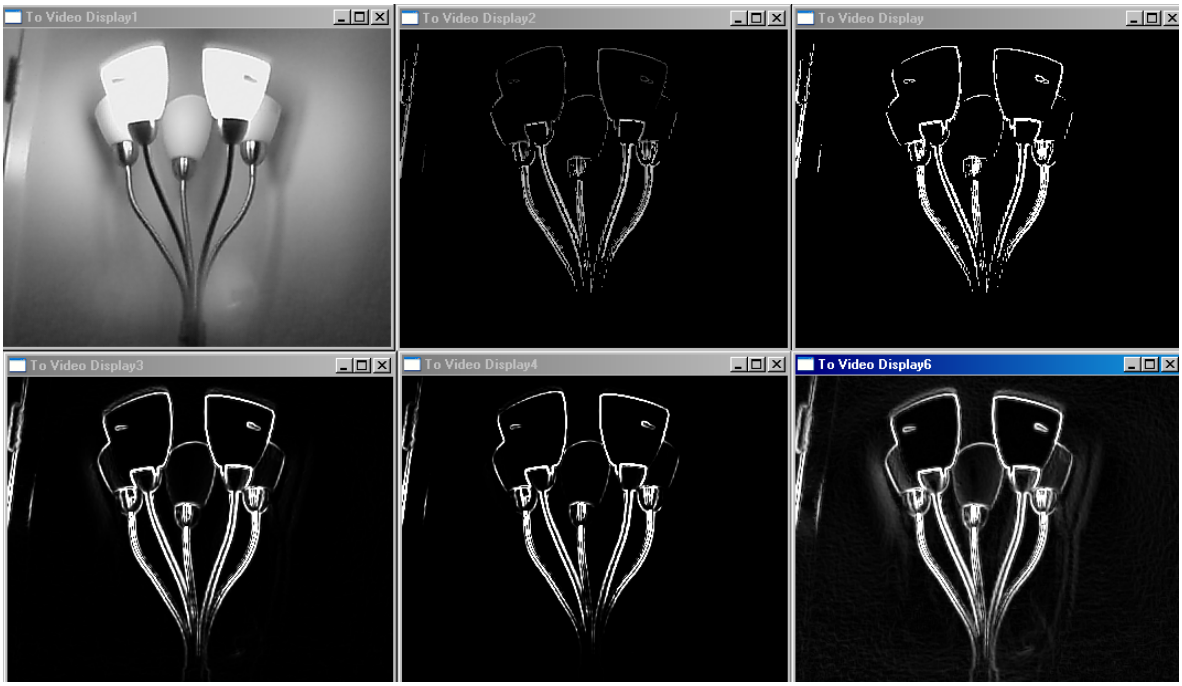

Figure 11: Simulation Results

The Digital Communication System Design project

This project familiarizes students with knowledge, tools, algorithms, and implementation platforms needed to design a communication (transceiver) subsystem. Students will be exposed to a variety of design options such as precision, quantization, sampling, and bit error rate (BER). The subsystems include matching filters, digital modulation (PSK, QAM, 4-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK), up/down conversion, DAC, timing and simulation, data recovery, demodulation, and BER testing. To prepare students for this project, the following set of laboratory exercises, with varying complexities, was developed. These exercises guide students in designing the components of telecommunication systems and implementation of the systems using an FPGA board. The exercise set includes the following:

- Design a matched filter, a polyphase filter for a given specification.
- Use MATLAB's bertool to compare the theoretical BER performance of various PSK and QAM modulation schemes.
- Build a BER testing model in Simulink, using blocks from the Communications and Signal Processing blocksets. Simulate the BER performance of 4-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK for a subset of signal-to-noise ratio (SNR) values.
- Design a Simulink demodulator.
- Build a system generator model that implements the 4-QAM, 16-QAM, 64-QAM, 8-PSK, and 16-PSK modulation schemes. Apply a single input of binary data bits running at a fixed sample rate to all modulation schemes. Compare the results with those obtained from the Simulink model.
- Parameterize the modulation schemes.
- Design a parameterized up-sample, and interpolate up-converter/down-converter, and digital/analog converter (DAC) units for a specific symbol rate, interpolation rate, carrier frequency, and output sample rate.
- Design a carrier recovery system using a phase error detector and a loop filter.
- Implement a timing synchronization scheme between the transmitter and receiver units.
- Generate the VHDL code for the transceiver system, and implement the transceiver using the FPGA board. Test the system.

After completing these exercises, students will posses the technical skills needed to design and simulate any type of digital transmitter/receiver communication system.

Major Challenges and Results

Introducing system-level and embedded processing design tools into undergraduate courses presents major challenges to the instructors of these courses. These include the following challenges:

a) Number and functionalities of these tools: A large number of system-level and embedded processing tools and environments are used. These tools are produced by MathWorks, Xilinx, and Mentor Graphics. These tools are used for design creation; functional and timing simulation; debugging; design synthesis; and to map, route, and download the design onto the implementation FPGA device.

b) Documentation sizes: Most of these tools have lengthy user guides, references, and getting started manuals.
c) Frequency of tools updates and modifications: Currently, most of the software tools are being updated twice each year. FPGA chips and boards become obsolete in a few years.

The above mentioned challenges represent some of the major problems facing the instructor of courses that use these design tools. It is extremely time consuming to develop good tutorials and meaningful laboratory exercises and senior design projects. Moreover, due to the high frequency of software updates and hardware advances, the developed tutorials and exercises may need to be updated each semester. In addition, instructors need to spend large amounts of time to help students with their work with these tools and to reduce their levels of anxiety and frustration.

Student Feedback and Conclusions

The authors have been working on developing system-level design projects for the past few years. So far, student course evaluations have shown extremely positive responses to the materials introduced in previous capstone and digital design classes. Analysis of student responses to these newly-developed materials will be the subject of another paper.

References

[1] *PicoBlaze user Guide*, Xilinx Inc., http://www.xilinx.com/userguids/ug129.pdf.
[2] *MicroBlaze Processor Reference Guide*, Xilinx Inc., http://www.xilinx.com/ise/embedded/edk92i_docs/mb_ref_guide.pdf.
[3] *EDK Concepts, Tools, and Techniques—A Hands-on Guide to Effective Embedded System Design*, Xilinx Inc., http://www.xilinx.com/ise/embedded/edk92i_docs/edf_ctt.pdf.
[4] *IBM PowerPC ISS Reference Guide*, Xilinx Inc., http://www.xilinx.com/ise/embedded/edk92i_docs/oslib_rm.pdf.
[5] Rangarajan, Srikanth. "Algorithms for Edge Detection." Stony Brook University. Web document link: www.ee.sunysb.edu/~cvl/ese558/s2005/Reports/Srikanth%20Rangarajan/submission.doc.
[6] Mohsen Sharifi, Mahmoud Fathy, Maryam Tayefeh Mahmoudi. "A Classified and Comparative Study of Edge Detection Algorithms." 2002 IEEE International Conference on Information Technology, Las Vegas, NV, January 2002, PP. 117–120.