AC 2010-1166: TOWARD AN INTERACTIVE ENVIRONMENT FOR EMBEDDED SYSTEMS DESIGN

Fadi Obeidat, Virginia Commonwealth University

Fadi Obeidat is a Ph.D. candidate in electrical and computer engineering dept. at Virginia Commonwealth University. He received her B.S. and M.S. degrees in Computer Engineering from Jordan University of Science and Technology and Yarmouk University, respectively. His research focuses on Embedded Systems Design, Performance Modeling and Computer Architecture.

Ruba Alkhasawneh, Virginia Commonwealth University

Ruba A. Alkhasawneh is a Ph.D. student in electrical and computer engineering dept. at Virginia Commonwealth University. She received her B.S. and M.S. degrees in Computer Engineering from Jordan University of Science and Technology and Yarmouk University, respectively. Her research focuses on diversity issues and engineering education.

Jerry Tucker, Virginia Commonwealth University

Dr. Tucker received his BSEE from the Mississippi State University in 1963 at which time he joined NASA Langley Research Center. While at Langley Research Center he received his MEEE in 1969 from the University of Virginia and in 1974 his Ph.D. in Electrical engineering from Virginia Tech. Dr. Tucker is currently an Associate Professor of Electrical Engineering at the Virginia Commonwealth University in Richmond, VA. His research interests include: Computer architecture, parallel processing, embedded microprocessor system hardware and software, VHDL based FPGA design, reconfigurable logic, ASIC design, Boolean equations, and Boolean calculus.

Robert Klenke, Virginia Commonwealth University

Robert H. Klenke is currently an Associate Professor of Electrical Engineering at the Virginia Commonwealth University. His research interests include system level modeling, hardware description languages, and unmanned aerial vehicle (UAV) flight control systems and applications. Dr. Klenke received his B.S. degree in Electrical Engineering from the Virginia Military Institute in 1982, and his M.S. and Ph.D. Degrees in Electrical Engineering from the University of Virginia in 1989 and 1993, respectively. He is a Senior member of the IEEE and a member of the IEEE Computer Society, Tau Beta Pi, and Eta Kappa Nu.

Toward an Interactive Environment for Embedded Systems Design

Abstract

In this paper, we propose building an interactive environment for embedded systems design using Nexys2 board from Digilent where a MicroBlaze soft-core processor and a VHDL monitor interface have been configured on the Xilinx Spartan-3E FPGA. This infrastructure allows an easy integration of hardware/software modules and a flexible monitoring for application's signals-of-interest, which in turn, enables students enrolled in an embedded systems class to interact directly with software and hardware components via monitor interface allowing an interactive debugging for the system-under-development. Moreover, as an implementation of problem based learning in engineering education, the project itself is a practical implementation of an embedded system that aims to walkthrough basic skills needed in embedded systems design.

Introduction

Field Programmable Gate Arrays (FPGAs) have been used in many embedded applications due to their ever-increasing level of performance, low cost, and re-configurability. For example, FPGAs have been used to accelerate a wide range of applications where the applications' computation-intensive parts can be implemented in hardware (on FPGA)[1-3]. Available gate count per FPGA chip has reached numbers that allow for implementation of very complex applications with the ability to implement soft-core processors such as MocroBlaze (from Xilinx)[4] and Nios-II (from Altera)[5], which in turn form a fertile environment for hardware/software co-design. In general, embedded systems work with limited resources (e.g., memory and power) in a real-time environment by employing a combination of software (SW) and hardware (HW) resources.

During the last couple of decades, industry needs have increased for embedded system engineers who possess both HW design and SW programming skills[6,7]. Hence, embedded systems design, as a topic, has been recently adopted by universities as one of the undergraduate/graduate courses/majors in the computer engineering area. Students enrolled in these courses are assumed to have a background in programming and hardware design skills using assembly languages, C, and hardware description languages (HDL) such as VHDL. Efforts have been made to define a set of theoretical and practical educational methodologies that help in achieving better outcomes of such courses[8-13]. In 2005, a workshop for embedded system education was held in conjunction with EMSOFT embedded software conference[14]. The presented papers discussed three main factors that affect the educational process in the embedded systems field: 1) teaching experience, 2) curricula and contents, and 3) labs and platforms. For example, the importance of enhancing the laboratory environment for improving embedded systems education process is shown in [12]. This work points to the significant role of using current available technologies and tools such as hard/soft-core processors, IP (Intellectual Property) cores, and the EDK (embedded development kit) tool in embedded systems labs. It also shows the importance of transition from using TTL ICs (transistor–transistor-logic integrated circuits) to reconfigurable devices such as FPGA. In [13] a set of experiments are proposed to enable students to acquire a set of practical skills

needed in HW/SW co-design. The experiments focus on developing embedded software (using C), FPGA design (using VHDL), and HW/SW co-design for the JPEG (Joint Photographic Experts Group) encoder.

In this paper, we concentrate on enhancing the laboratory environment (practical side) for improving embedded systems design skills by proposing a project to walkthrough the basic skills needed in embedded systems design such as: accessing memory/peripherals, adopting IP cores, handling interfacing and synchronization issues, and developing HW/SW co-design skills. The project itself is a practical implementation of an embedded system using Nexys2-500 board from Digilent[15]. The project aims to prepare an interactive design environment in which computer engineering students will have an infrastructure to learn, build, and develop embedded system applications, and directly interact with SW/HW components of the system-under-development. The infrastructure consists mainly of a MicroBlaze soft-core processor and a VHDL monitor interface that have been configured on the Xilinx Spartan-3E FPGA, which in turn, allow an easy integration of HW/SW modules and visualizing an application's signals-of-interest via a VGA (Video Graphics Array) monitor interface – i.e. the designer is able to trace any internal signal (or intermediate results) in his design at real time.

In addition to the skills gained from implementing such infrastructure, this environment eliminates the use of a serial UART connection for assisting in system debugging. While UART is flexible to use, we believe that building or integrating a VHDL monitor interface to trace signals-of-interest covers a set of important skills that enable new students to be familiar with FPGA design such as handling clock gating issues, utilizing FPGA memory resources (BlockRAM) and practicing hardware design intensively using HDL. To illustrate the flexibility of this infrastructure, two applications have been implemented, the first one is a VHDL implementation of a frequency counter, and the second one is a VHDL/C implementation for measuring program execution time.

The remaining of this paper is organized as follows: the components and resources used in this project are described first. Then, the main modules, dataflow and application examples are explained in the design and implementation section. After that, the main educational aspects of developing this infrastructure are discussed. Finally, the conclusions and future work are presented.

Components and Resources

In this section, we provide an overview about the main resources/tools employed to build the proposed infrastructure:

- *Nexys2-500 Board from Digilent*[15]: consists of the Xilinx Spartan-3E 500K FPGA equipped with many input/output (I/O) peripherals (see Fig.1). A USB (Universal Serial Bus) port is used for providing board power, programming, and data transfers. The FPGA is also connected to I/O devices such as push buttons, switches, LEDs (light-emitting diodes) and seven-segment-displays. Data ports like keyboard/mouse PS/2, monitor VGA, RS232 are equipped and hardwired with FPGA predefined pins. In addition to that, a 16-MByte Flash memory and a 16-MByte SDRAM are connected for memory demanding applications.

Nexys2 board has a 50MHz oscillator which works as a system clock. It has 60 FPGA I/O pins routed to expansion connectors which can be configured by the user. The Xilinx EDK tool can be used for design and configuration purposes.

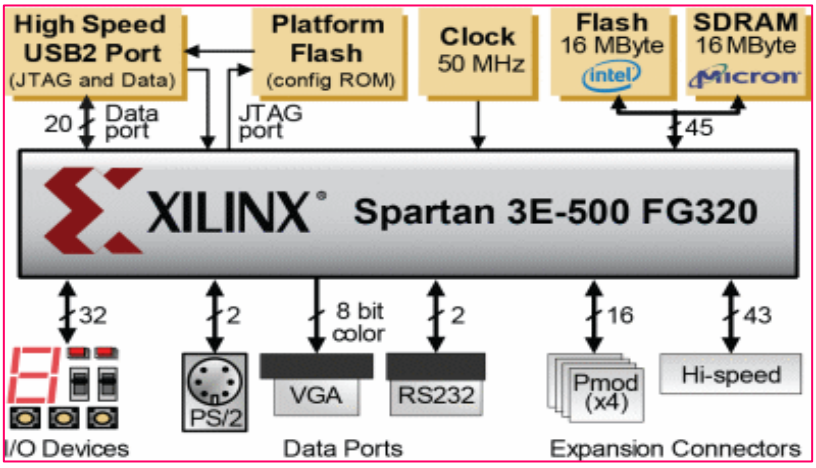

Figure 1. Digilent Nexys2 Board Layout[15]

- *Xilinx Spartan 3E-500K FPGA*[16]*:* consists of 1,164 configurable logic blocks (CLBs), 360K block RAM bits, 20 dedicated multipliers, 4 DCMs, and 232 I/O pins. With system gates count of 500K, MicroBlaze soft-core processor can be configured on this FPGA with the capability of using the remaining FPGA resources for implementing HW custom designs.

- *MicroBlaze*[17]*:* is a soft-core processor which can interact with FPGA resources through *processor local bus* (PLB) bus (see Fig.2). MicroBlaze has the following features:

    - Reduced instruction set computer (RISC)
    - 32-bit general purpose registers
    - 32-bit instruction word with three operands and two addressing modes
    - 32-bit address bus

  MicroBlaze has an arithmetic logic unit (ALU) and a shift functional unit with the option of adding barrel shift, multiplier, divider, and/or floating point functional units. Data cache, instruction cache, memory management unit are also optional configurable units. MicroBlaze can be configured to consist of a three stage pipeline (customized area: fetch, decode, and execute) or a five stage pipeline (fetch, decode, execute, access-memory and write-back). Programming an application to run on the MicroBlaze platform can be either at the assembly level or C level. An application can work as stand-alone application or run under an embedded operating system such as uClinux.

- *The Embedded Development Kit (EDK)*[19]*:* a tool provided by Xilinx allows integrating both hardware and software modules for application development and implementation, i.e., it allows the user to write/compile HDL modules and C applications to be downloaded on the FPGA/ MicroBlaze platform. EDK offers many IP cores to be used based upon the application requirements.

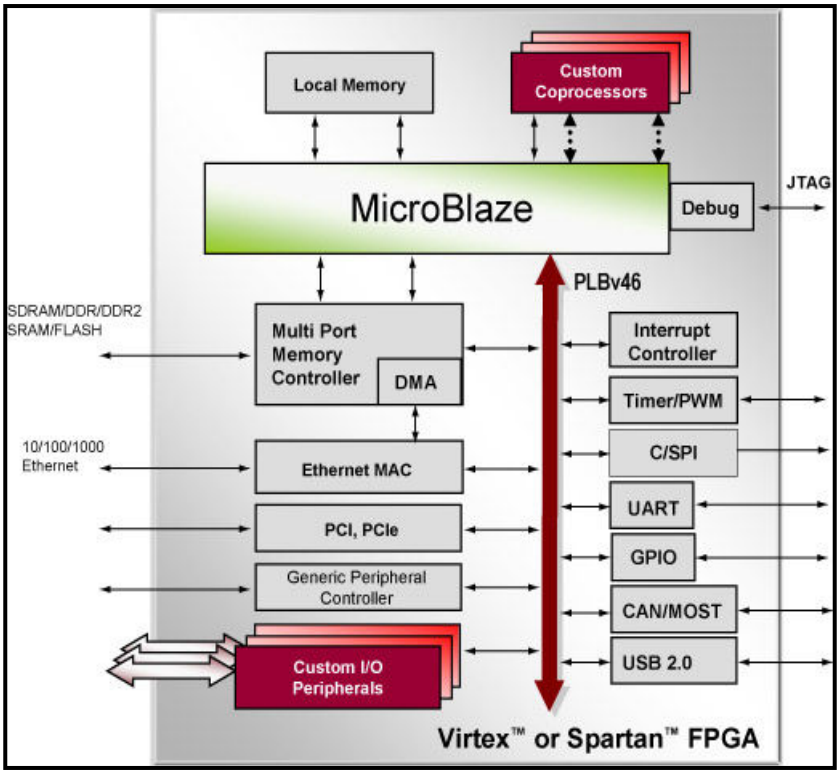

Figure 2. Soft-Core MicroBlaze System Interface[18]

Design and Implementation:

System Overview:

A set of 20 registers have been configured to be accessed by both HW (HDL modules) and SW (C-applications running on MicroBlaze) via PLB bus. For monitoring signals-of-interest of an application, a set of 300 registers have been predefined in VHDL and mapped into unique locations on the monitor. Hence, the designer can assign the signals-of-interest of his/her application to these registers and know where exactly the content of each register is going to be displayed on the monitor. This also allows for more data representation, in terms of characters' shapes and quantity on an output device compared to the limited number of units of the seven-segment-displays available on the board. The user has the option to control which signal assigned to which register either at the design time (so each time the user changes the signal-of-interest for a certain register, he/she should go through the whole FPGA design process – compiling, synthesizing, placing and routing), or at run time by using a multiplexing technique by which each register can be connected to more than one signal via a multiplexer where the selection lines of the multiplexor can be set through the software running on the MicroBlaze (via shared registers) or by interfacing the on-board switches or buttons equipped using HDL module.

This system consists of the following main modules (see Fig.3):

- Monitor Interface

- ROM (read only memory)
- RAM (random access memory)
- Character Converter
- Two application examples:
  - Frequency Counter
  - Program Cycle Counter

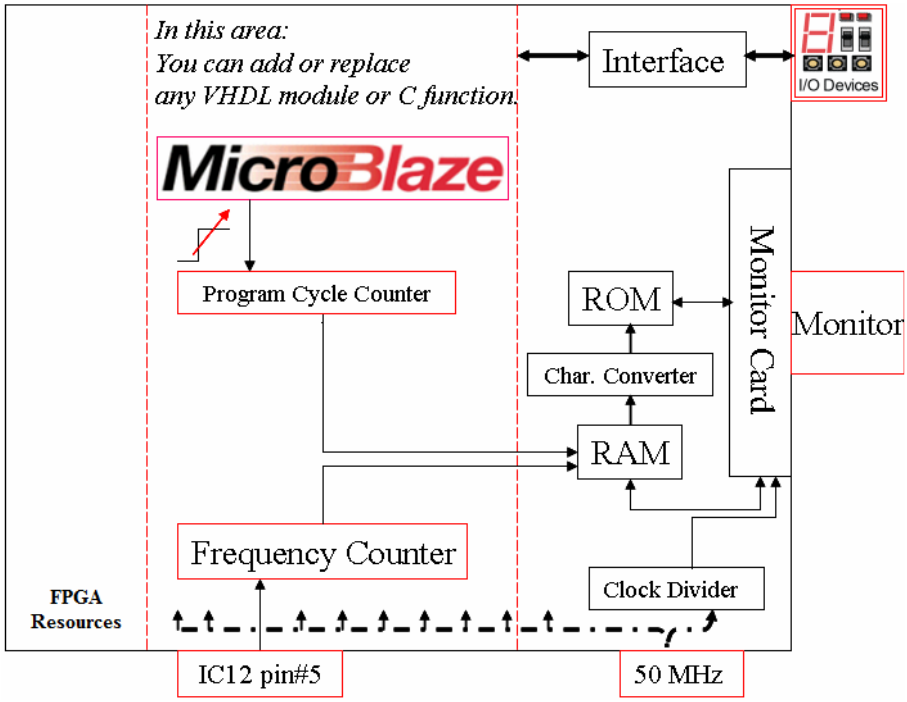

Figure 3. Project Block Diagram

Data Flow:

The monitor interface module generates horizontal and vertical synchronization signals to scan the monitor via the VGA connector line by line, pixel by pixel displaying 60 frames/second. Simultaneously, the monitor interface turns these pixels on/off based on the shape of the character being scanned from the memory. The monitor interface is also responsible to set the font and the background colors by passing a set of 8 signals to the VGA *red-green-blue* (RGB) pins. Digilent attached 8 different resistors to the RGB pins allowing getting a total of 256 colors[15]. These signals have been assigned to the on-board switches (by a VHDL module) to allow the user controlling the font and the background colors by changing the switches combination.

The ROM module has been implemented using the Xilinx System Generator tool as a *.coe* file which can be set as read only memory. A total of 64 characters (shapes) are stored in the ROM as 8*8 bits format. Each character can be accessed by specifying its unique starting address and contents coordinates to select which pixel of the character is going to be scanned/displayed.

The RAM module has been built using the *Look-Up-Table* (LUT) technique. RAM is used to store the characters required to be displayed on the monitor. Each byte of the LUT is attached to a certain location on the monitor. So, the user can select at which location certain results will be displayed by assigning these results to the corresponding bytes on the LUT.

The monitor interface, ROM, and RAM modules all work synchronously. While monitor interface scans the monitor, it sends the RAM module the coordinates of the currently displayed digit. Based on that, the RAM sends the stored value (signal-of-interest) that is pre-assigned to that location to the character convertor module which in turn maps this value to its character format in the ROM. Simultaneously, the ROM receives the coordinates of the exact pixel being displayed, based on which, it sends the monitor interface the content of that pixel immediately which turns the corresponding monitor pixel on/off.

Application examples:

Two applications have been implemented to show the system's capability and flexibility. The first application was building a frequency counter to measure the frequency of a guest signal attached to an external pin of the FPGA (see Fig.4). The frequency counter is a VHDL module counting in a decimal manner. The counter is incremented every positive edge of the incoming signal and cleared every one second, keeping last measurement valid for the whole incoming second. The result (frequency measurement) is stored as eight decimal digits in a set of registers that are attached to a sequence of eight locations in the RAM allowing displaying the measurement in Hz on pre-assigned locations on the monitor. The frequency counter module can be also used to measure the frequency of any internal generated signal, which in turn enables the user to verify whether the design works properly or not. For example, tasks such as clock divider and frequency synthesizer can be assigned to the students and then students can use the frequency counter module for verification purposes.

The second application was implementing a program cycle counter to measure the number of cycles required to run certain application or function on the MicroBlaze (see Fig.5). This application shows the ability of HW/SW interaction via shared resources such as registers. The application has two parts. In the first part (the C part) a flag is set at the beginning of the target function and cleared at the end of this function. The flag is an accessible register by the HW/SW via PLB. In the second part (the VHDL part), a counter starts counting at the positive edge of the flag and ends counting at the negative edge of the flag. Counting process runs at the system clock speed, which is 50MHz in this case. The counter measurement, in hexadecimal format, is stored in registers attached directly to the RAM module which is periodically scanned by monitor interface to display its contents on the monitor. A 32-bit hexadecimal-to-decimal function can be assigned to students to display the measurements in the decimal format. Also, students can be asked to extract the communication overhead between the HW and SW via PLB.

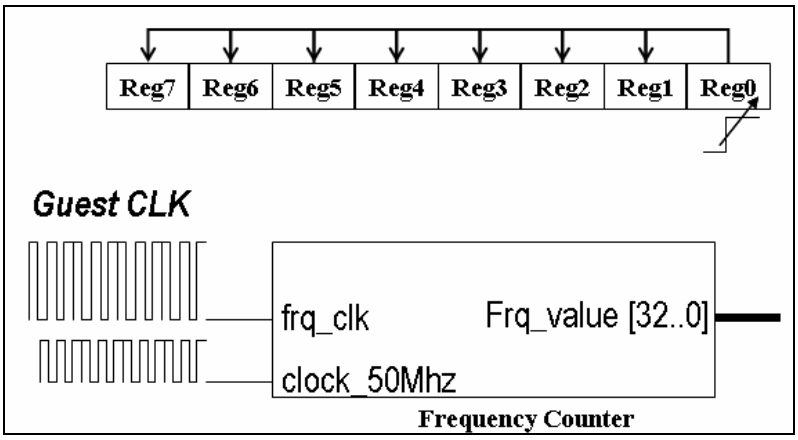

Figure 4. Frequency Counter Module

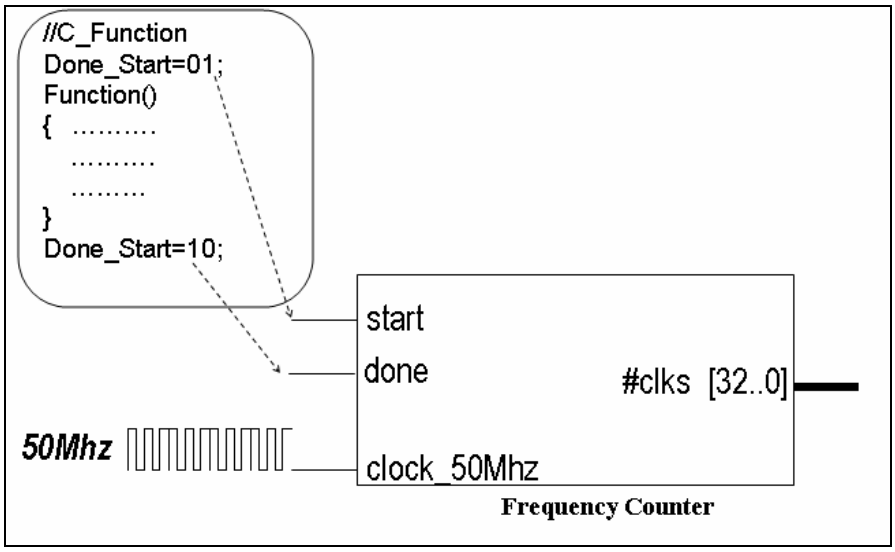

Figure 5. Program Cycle Counter Module

Both of the above examples show an alternative technique of using a logic analyzer for same purposes. In addition to flexibility of changing the signal-of-interest for measuring the frequency using the multiplexing technique, user saves time and cost for setting up equipments and connections between the processor and the logic analyzer.

Discussion

The proposed project outlines a framework for embedded systems education. The project can be exploited at two levels: application (user) level and system level. At application level, the infrastructure can be given to students along with a user guide asking them to implement specific tasks (mini-projects) where students will not go through the internal details of the system (e.g., the system level synchronization and interfacing issues), instead, they will have a set of predefined components that enable HW/SW communication via shared registers and passing the signals-of-interest to a set of registers that are pre-assigned to specific location on the monitor.

Hence the user can insert his/her own HDL/C design (see Fig.3) and visualize the results on the monitor. In the previous section, two examples have been shown of how to use this infrastructure for implementing a HW-based frequency counter and HW/SW-based program cycle counter, respectively. Other tasks can be assigned to students such as implementing a mathematical algorithm in VHDL and validate the results on HW using this infrastructure.

On the other hand, at system level, the system's main modules such as the monitor interface, ROM, and RAM along with their input/output and functionality specifications can be given to students asking them to integrate these parts and building similar infrastructures with a minimum of direction. Mini-tasks such as implementing a VHDL clock divider or implementing C-level calculator (and showing the results on monitor via a VHDL interface) can be also assigned to widen the scope of the students' understanding of the system's requirements and design. From this perspective, students will be able to develop and gain a set of basic skills needed in embedded systems design and digital design using HDL. For example, integrating the basic modules of this infrastructure provides an opportunity to learn how handle the communications between HW components using a top-level module. Moreover, skills like decoding, multiplexing, synchronization and interfacing are also considered very important in other computer engineering classes such as digital logic design and computer organization.

Overall, this project can be classified with the efforts of employing the *problem based learning* (PBL) methodology for embedded systems education and computer engineering education in general[20-22], where students are given a problem with a minimum of resources and direction and asked to analyze and solve the problem.

Conclusions and Future Work

In this paper, we have shown a practical implementation of an interactive environment for embedded systems design. This environment allows new users to insert their own HDL/C applications and trace the signals-of-interest via monitor without spending time on solving the interfacing and synchronization issues which allows for easier understanding and faster development of example HW/SW applications. A frequency counter and program cycle counter example applications have been implemented to show the flexibility of using this system. On the other hand, working on building/integrating this infrastructure helps students to understand/acquire basic skills in the field of embedded systems such as: decoding, multiplexing, accessing memory and I/O peripherals, adopting IP cores, and handling interfacing and synchronization issues.

Planned future work includes integrating more components in this infrastructure such as sensors, actuators and digital/analog convertors to allow students to practice other skills in the field of embedded systems. We plan also to exploit this infrastructure to introduce hardware design concepts in other computer engineering courses such as digital logic design and computer architecture where the students will be able to implement these concepts directly on the hardware and visualize the results via the monitor.

References

1. Z. K. Baker and V. K. Prasanna. "Efficient hardware data mining with the Apriori algorithm on FPGAs". In *Proceedings of the 13th IEEE Symposium on Field-Programmable Custom Computing Machines*, 2005.
2. B. de Ruijsscher, G. N. Gaydadjiev, J. Lichtenauer, and E. Hendriks. "FPGA accelerator for real-time skin segmentation". In *Proceedings of the 2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia*, 2006.
3. B. Harris, A. C. Jacob, J. M. Lancaster, J. Buhler, and R. D. Chamberlain. "A banded Smith-Waterman FPGA accelerator for Mercury BLASTP". In *Proceedings of the 2007 International Conference on Field Programmable Logic and Applications*, 2007.
4. Xilinx MicroBlaze: *http://www.xilinx.com/tools/microblaze.htm*
5. Altera Nios-II *http://www.altera.com/products/ip/processors/nios2/ni2-index.html*
6. P. Marwedel, D. Gajski, E. De Kock, H. De Man, M. Sami & I. Soderquist, "Embedded systems education: how to teach the required skills?", *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis,* 2004
7. M. Anderson "Help Wanted: Embedded Engineers Why the United States is losing its edge in embedded systems", *Article: IEEE-USA Today's Engineer Online,* Feb 2008, *http://www.todaysengineer.org/archive/date.asp?long_dat=Feb+08*
8. D. Moeller & H. Vakilzadian, "Virtual Prototyping Methodology As A Replacement For Physical Design In Teaching Embedded Systems", *American Society for Engineering Educators 2009 Annual Conference*, Austin, TX, June 2009.
9. E. Montanez & M. Norman, "A Cost-Effective, Modular-Hardware Platform For Embedded Systems Design And Development", *American Society for Engineering Educators 2009 Annual Conference*, Austin, TX, June 2009.
10. J. Holt, H. Shi & H. Stern, "Educational Goals For Embedded Systems In The Multicore Era", *American Society for Engineering Educators 2009 Annual Conference*, Austin, TX, June 2009.
11. Steven Barrett, Jeffrey Anderson, Jerry Hamann, Robert Kubichek, Suresh Muknahallipatna, John Pierre, David Whitman & Cameron Wright, "Embedded Systems Design: Responding To The Challenge", *American Society for Engineering Educators 2009 Annual Conference*, Austin, TX, June 2009.
12. S. Merchant, G.D. Peterson & D. Bouldin, "Improving embedded systems education: laboratory enhancements using programmable systems on chip", In *Proceedings of the 2005 International Conference on Microelectronic Systems Education*, Anaheim, CA, USA, June 2005.
13. H. Mitsui*, H.* Kambe*, D.* Tilwaldi *& H.* Koizumi*,* "A Student Experiment Method for Embedded System Education Based on Incremental Upgrade", In *Proceedings of the 2007 International Conference on Parallel Processing Workshops ICPPW, 2007*.
14. D.J. Jackson & P. Caspi, "Embedded systems education: future directions, initiatives, and cooperation", ACM SIGBED Review, v.2 n.4, p.1-4, October 2005.
15. Digilent Nexys2 Board Reference Manual: *http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf*
16. Spartan-3E FPGA Family: Complete data Sheet: *http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf*
17. MicroBlaze Processor Reference Guide: *http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf*
18. MicroBlaze System Interface *http://www.eefocus.com/data/08-05/4155_1211284808/1211286716.jpg*
19. EDK concepts, tools, and techniques: *http://www.xilinx.com/support/documentation/sw_manuals/edk_ctt.pdf*
20. A. Striegel and D. T. Rover. "Problem-based learning in an introductory computer engineering course". In *Proc. of the 32nd ASEE/IEEE Frontiers in Education Conference, November, 2002.*
21. J. Kay and B. Kummerfeld. "A problem-based interface design and programming course". In *Proc. of SIGSCE, 1998.*
22. M. Garcia-Famoso. "Problem-based learning: a case study in computer science". In Recent Research Developments in Learning Technologies, 2005.