11-2004

System–on–a–programmable–chip development platforms in the classroom

Tyson S. Hall
Southern Adventist University, tyson@southern.edu

James O. Hamblen
Georgia Institute of Technology - Main Campus

Recommended Citation

T. S. Hall and J. O. Hamblen, "System–on–a–programmable–chip development platforms in the classroom,"IEEE Transactions on Education, vol. 47, no. 4, pp. 502–507, Nov. 2004.

System-on-a-Programmable-Chip Development Platforms in the Classroom

Tyson S. Hall and James O. Hamblen

***Abstract*—This paper describes the authors' experiences using a system-on-a-programmable-chip (SOPC) approach to support the development of design projects for upper-level undergraduate students in their electrical and computer engineering curriculum. Commercial field-programmable gate-array (FPGA)-based SOPC development boards with reduced instruction set computer (RISC) processor cores are used to support a wide variety of student design projects. A top-down rapid prototyping approach with commercial FPGA computer-aided design tools, a C compiler targeted for the RISC soft-processor core, and a large FPGA with memory is used and reused to support a wide variety of student projects.**

***Index Terms*—Altera, field-programmable gate array (FPGA), microblaze, Nios, processor core, system on a chip (SOC), system on a programmable chip (SOPC), Xilinx.**

I. Introduction

A new technology has emerged that enables designers to utilize a large field-programmable gate array (FPGA) that contains both memory and logic elements along with an intellectual property (IP) processor core to implement a computer and custom hardware for system-on-a-chip (SOC) applications. This new approach has been termed system-on-a-programmable-chip (SOPC). During the past two years, several commercial reduced instruction set computer (RISC) processor cores have been introduced [1]. In this paper, the authors will overview several commercial processor cores that can be used in the classroom, explore the computer-aided design (CAD) tool flow involved with this process, and highlight sample student projects that have used this technology.

II. Technology Overview

A. SOPC Processor Cores

Hard-processor cores use an embedded-processor core (in dedicated silicon) in addition to the FPGA's normal logic elements. Hard-processor cores added to an FPGA are a hybrid approach, offering performance tradeoffs that fall somewhere between a traditional application-specific integrated circuit (ASIC) and an FPGA; they are available from several manufacturers with a number of different processor flavors. For example, Altera offers an ARM processor core embedded in its APEX 20KE family of FPGAs that is marketed as an Excalibur device. Xilinx's Virtex-II Pro family of FPGAs include up to four PowerPC processor cores on-chip. Cypress Semiconductor also offers a variation of the SOPC system. Cypress's Programmable-System-on-a-Chip (PSoC) is formed on an M8C processor core with configurable logic blocks designed to implement the peripheral interfaces, which include analog-to-digital converters, digital-to-analog converters, timers, counters, and universal asynchronous receivers–transmitters (UARTs) [2], [3].

Soft cores, such as Altera's Nios and Xilinx's MicroBlaze processors, use existing programmable logic elements from the FPGA to implement the processor logic. As seen in Table I, soft-core processors can be very feature rich and flexible, often allowing the designer to specify the data path width, the automatic logic unit (ALU) functionality, number, and types of peripherals, and memory-address space parameters at compile time. However, such flexibility comes at a cost. Soft cores have slower clock rates and use more power than an equivalent hard-processor core.

TABLE I
Features of Commercial Soft Processor Cores

Feature	Nios 3.1	MicroBlaze 3.2
Datapath	16 or 32 bits	32 bits
Pipeline Stages	5	3
Frequency	up to 150 MHz[1]	up to 150 MHz[1]
Gate Count	26,000–40,000	30,000–40,000
Register File	up to 512	32 general purpose
	(window size: 32)	and 32 special purpose
Instruction Word	16 bits	32 bits
Instruction Cache	Optional	Optional
Hardware Multiplier	Optional	Optional

With current pricing on large FPGAs, the addition of a soft-processor core only costs a few dollars based on the logic elements it requires. The remainder of the FPGA's logic elements can be used to build application-specific system hardware. Traditional SOC devices [ASICs and custom very-large-scale-integration integrated circuits (VLSI ICs)] still offer higher performance, but they also have large development costs and longer turnaround times [4]. For student projects requiring an actual hardware implementation, the FPGA-based SOPC approach is easier, faster, smaller, and more economical.

Typically, additional software tools are provided along with each processor core to support SOPC development. A special computer-aided design (CAD) tool specific to each soft-processor core is used to configure processor options, which can include register file size, hardware multiply and divide, interrupts, and input/output (I/O) hardware. This tool outputs a hardware description language (HDL) synthesis model of the processor core in very-high high-speed integrated circuit hardware description language (VHDL) or Verilog. In addition to the processor, other system logic is added, and the resulting design is synthesized using a standard FPGA synthesis CAD tool. The embedded application program for the processor is typically written in C or C++ and compiled using a customized compiler provided with the processor core tools.

B. SOPC Development Hardware

SOPC boards and required CAD tools are available from both Altera and Xilinx [5], [6]. The Altera Nios development board shown in Fig. 1 was one of the earliest SOPC boards available. It contains a 200K-gate FPGA, Flash, and SRAM memory on-board, as well as several I/O options and connectors for attaching external devices. The development kit includes a complete set of tools for SOPC design.

A number of daughter cards are available for this board to extend its functionality. For projects that require networking, a custom Ethernet kit is available. A custom CompactFlash board can also be added if additional and/or removable storage is needed. In addition, third-party vendors make a number of add-on boards that can be interfaced directly to the Nios board via its standard PCI Mezzanine Connector (PMC).

Manuscript received April 23, 2003; revised November 10, 2003.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

Digital Object Identifier 10.1109/TE.2004.825926

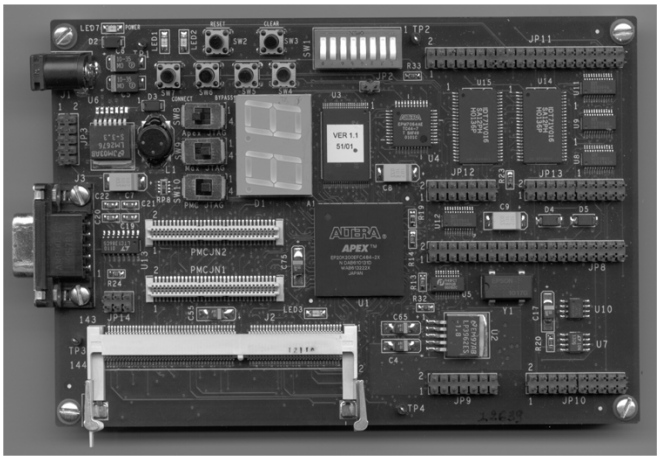

Fig. 1. Altera's Nios board contains a 200 000 gate FPGA, Flash, SRAM, several I/O options, and a RISC soft-processor core for the FPGA.

Several third-party vendors also provide software to aid in the development of systems on the Nios processor. Everything from real-time operating systems to advanced debugging tools are available. Of particular interest, there is a μ Clinux kernel that runs on the board, but licensing fees still make this add-on kit somewhat expensive for student projects.

Fig. 2 shows a low-cost Digilent 2E FPGA board that contains a 200 000 gate Xilinx Spartan-IIE FPGA [7]. This board can be used for SOPC development with Xilinx's MicroBlaze processor core. This particular board has very limited functionality outside of the FPGA (no external memory, high-speed conntectors, etc.); however, it is very economically priced (about one fifth of the cost of a Nios board) and can be an attractive option for student projects. To support the SOPC projects, an additional memory module was designed as a student project and attached to the board's header connectors. The additional memory is needed to support the development of larger programs.

Several daughter cards are available from Digilent to extend the functionality of this board including Ethernet, USB, parallel port, serial port, analog I/O, and digital I/O boards. In addition, with three 40-pin, general-purpose I/O headers, this board is designed to act as a system board with project-specific functionality added via custom peripheral boards.

III. SOPC Design Using CAD Tools and FPGAs

A. Traditional Tool Flow

The traditional flow of commercial CAD tools typically follows a path from hardware description language (HDL) or schematic design entry through synthesis and place and route tools to the programming of the FPGA. FPGA manufacturers provide CAD tools such as Altera's Quartus II and Xilinx's ISE software, which take the designer through this process in steps. As shown in Fig. 3, the addition of a processor core and the tools associated with it are a superset of the traditional tools. The standard synthesis, place and route, and programming functionality are still needed, and in the case of both Altera and Xilinx, the same CAD tools (Quartus II or ISE) are used to implement these blocks.

B. Processor Core Configuration Tools

Today, a number of predefined processor cores are available from various sources. General public licensed (GPL) processor cores can be found on the Web (i.e., www.opencores.org), while companies such as Altera (Nios processor), Xilinx (MicroBlaze processor), and Tensilica (Xtensa processor) provide their processors for a fee. This paper will focus on the processors provided by the FPGA manufacturers, although cores from third-party sources are similar in nature.

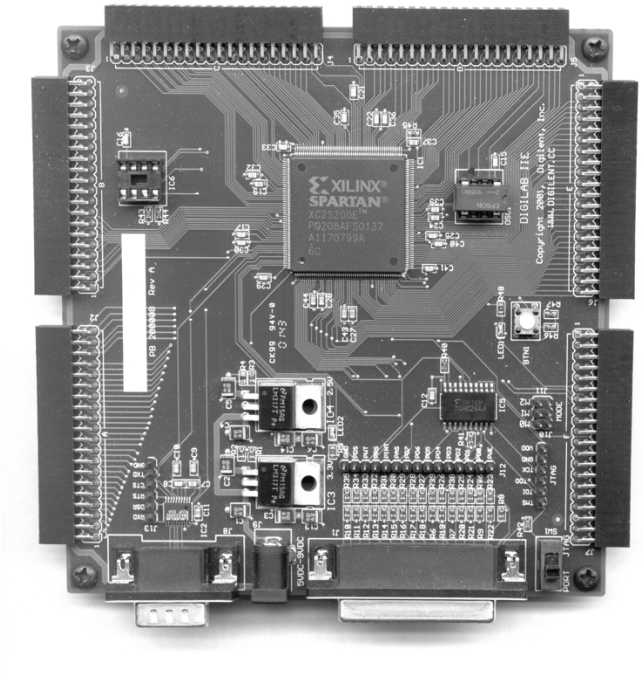

Fig. 2. Digilent's Digilab 2E low-cost FPGA board contains a 200 000 gate Xilinx FPGA that can support Xilinx's MicroBlaze soft core.

Processor cores provided by FPGA manufacturers are typically manually optimized for the specific FPGA family being used and, as such, are more efficiently implemented on the FPGA than a student-designed core (especially given the time and resource constraints of most class projects). in addition, these companies provide extensive support tools to ease the customization and use of their cores, including high-level compilers targeted at the custom cores (see Section III-C).

In the case of Altera and Xilinx, the Processor Core Configuration Tool block shown in Fig. 3 is realized in a user-friendly graphical user interface (GUI) that allows the designer to customize the processor for a particular project. The configurable parameters include the data path width, memory, address space, and peripherals (including arbitrarily defined general-purpose I/O, UARTs, Ethernet controllers, and memory controllers). Once the processor parameters are specified in the GUI interface, the processor core is generated in the form of an obfuscated HDL file (in Altera) or a netlist file (in Xilinx). This file can then be included within a traditional HDL design using the standard CAD tools. Specific pin assignments and additional user logic can be included at this point like any other FPGA design. Next, the full hardware design (processor core and any additional user logic) is compiled (synthesis, place and route, etc.), and the FPGA can be programmed with the resulting file using the standard tools. At this point, the hardware design is complete, and the FPGA logic has been determined.

C. High-Level Compiler for Processor Core

As shown on the right-hand side of Fig. 3, the next step is to write and compile the software that will be executed on the soft-processor core. When the Processor Core Configuration Tool generates the HDL or netlist files, it also creates a number of library files and their associated C header files that are customized for the specific processor core generated. A C/C++ compiler targeted at this processor is also provided. The designer can then program stand-alone programs to run on the processor. As an option, the designer can compile code for an operating system targeted for the processor core. Altera sells an add-on

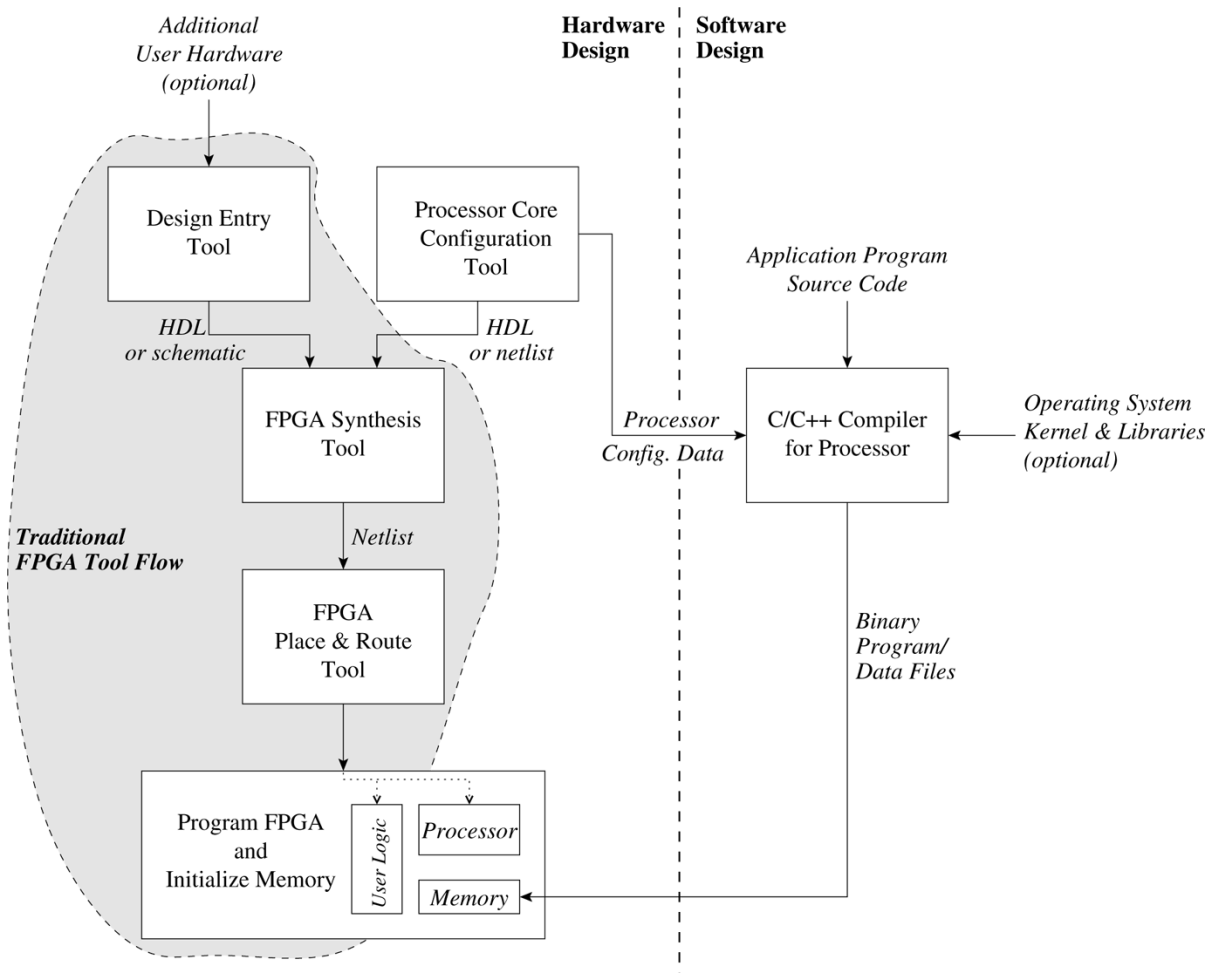

Fig. 3. CAD tool flow for SOPC design is comprised of the traditional design process for FPGA-based systems with the addition of the processor core configuration tool and software design tools. In this figure, the program and data memory is assumed to be on-chip for simplicity. Fig. 4 shows a more realistic memory configuration with external memory.

kit that includes a version of μ Clinux that has been ported to the Nios processor, and several other operating systems are available from third-party vendors.

D. Initializing Memory

Once a program/data binary file has been generated, it must be loaded into the processor's program and/or data memories. This loading can be done in several ways depending on the memory configuration of the processor at hand.

1) On-Chip Memory: If the application program is small and can fit into the memory blocks available on the FPGA, then the program can be initialized in the memory when the hardware configuration is programmed (see Sections III-A). This initialization is done through the standard FPGA tools, such as Altera's Quartus II software or Xilinx's ISE software. However, on-chip memory is typically very limited, and this solution is not usually an option.

2) Bootloader: In a prototyping environment, the application program will most likely be modified a number of times before the final program is complete. In this case, one can load a "bootloader" program into the on-chip memory that starts running on boot-up. This program is small enough to fit in most on-chip memories, and its primary function is to receive a program binary file over the serial port (or other interface), load it into external memory, and then start the new code executing. In this way, a new program can be stored into external memory (SRAM, SDRAM, Flash, etc.) by downloading it over the serial port (or other interface) on the fly without having to reload the FPGA's hardware configuration.

Altera includes a bootloader with its Nios processor called GERMS. GERMS provides a shell interface with limited debugging capabilities (view memory contents, erase memory locations, write to memory locations, etc.) in addition to the basic bootloader functionality. Xilinx provides a debugger called XMDstub that includes the ability to download a program binary over the serial port (or other interface), store it in memory, and start the code executing. However, depending on the type of external memory being used, the XMDstub source code may have to be modified to properly interface with the memory. In addition, the debugging functionality implemented in XMDstub can be removed to provide a simple bootloader that only provides the program download capabilities.

3) External Nonvolatile Storage: The application program code can be stored on an external EEPROM, Flash, or other form of nonvolatile memory. Either the application program can be preprogrammed in the external memory module (for a production run) or a bootloader program could be used to store the application program in nonvolatile storage. For low-speed applications, the code can be executed directly from the external memory. However, if high-speed functionality is required, then a designer could use three memories, as shown in Fig. 4. In this scheme, the on-chip memory is initialized with a bootloader, which handles the movement of the application

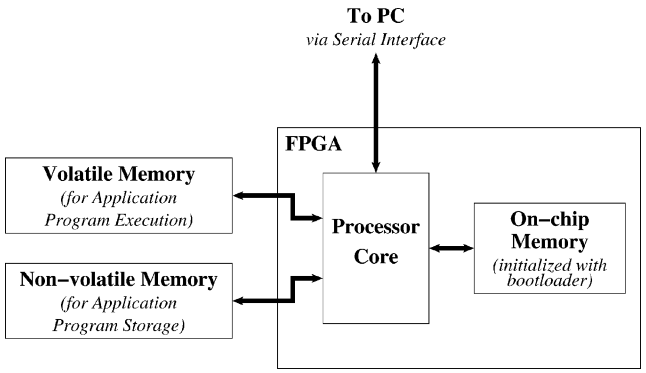

Fig. 4. This arrangement of on-chip and external memories provides flexiblity and performance to an SOPC system. The internal memory stores a bootloader program that can retrieve an application program from the nonvolatile memory or the PC via the serial interface and store it in volatile memory for execution. In addition, the nonvolatile memory can be initialized by the bootloader by storing an application program downloaded from the PC via the serial interface. Thus, fast execution times can be achieved by executing the program from high-speed SDRAM (volatile memory); permanent storage is afforded through the use of flash (nonvolatile memory); and flexibility in programming is achieved through the bootloader and serial interface.

program between the memories. The fast, volatile memory (i.e., SDRAM) is used to store the application program during execution. Finally, the slower, nonvolatile memory (i.e., Flash or EEPROM) is used for the permanent storage of the application program. The bootloader program can be modified to power-up, retreive a program from nonvolatile storage, store it in the faster, volatile memory, and then start it executing from the faster memory. This scheme provides the advantages of permanent storage, fast execution, and the ability to modify the application program when needed. Of course, it comes at the expense of having additional memory.

IV. Using SOPC in the Undergraduate Curriculum

For the past four semesters, the authors have used FPGA-based SOPC development boards to construct prototype systems for undergraduate student projects. SOPC boards present an interesting alternative to the more traditional commercial off-the-shelf microcontroller or basic FPGA board approach used to build student projects that require hardware and software, and their use has lead to a wide variety of successful student projects [8].

Based on experience with the existing SOPC tools, students need to have taken previous coursework in digital logic design, computer architecture, and C programming [9]–[11]. Some prior experience in VHDL or Verilog and exposure to FPGAs and their associated CAD tools is also useful. In most undergraduate curricula, these requirements will limit the application of SOPC designs to courses in the senior year.

There is still a significant learning curve to overcome when using these complex commercial CAD tools. Each new version of the SOPC CAD tools becomes easier to use, but they are still more complex than the basic FPGA CAD tools since more steps are required. Students still need some level of maturity and patience to make it through the complicated CAD tool flow for SOPC design. To help resolve this issue, students are now required to complete a system-level tutorial and demo, using a SOPC reference design during the first few weeks of any project course. This requirement forces them to start work earlier and to be familiar with the SOPC boards hardware and the complex CAD tool flow before the project-specific work starts.

The authors have found that an experienced user, such as a course instructor or teaching assistant, still needs to be available to install and maintain all of the CAD tools and help students when they occasionally encounter hardware and tool-related problems that they cannot resolve.

Fig. 5. Student project using the small hobbyist R/C Hummer vehicle with the color tracking CMUcam all controlled by the Altera Nios SOPC board.

A. Using SOPC in Senior Design

ECE 4006, Major Design Project, is an undergraduate, team-oriented design experience. It is a required three-hour semester course for both electrical and computer engineering students, normally taken by seniors. Students work together in teams of three or four on a semester-long design project. For computer engineering students, the design project must have both hardware and software elements and include engineering tradeoffs. A number of the student teams have used the SOPC approach to construct a prototype of their design. Projects have included Web servers, e-mail servers, vision systems, Internet appliances, and numerous robots [12]. In all of these projects, students have used the SOPC development tools to specify a soft-processor core and compile their embedded application program. They then use the traditional CAD tools to add any required custom hardware logic, compile the full system, and configure the FPGA.

Fig. 5 shows a student robotics project. An off-the-shelf hobbyist radio-controlled (R/C) vehicle was modified so that it is controlled by the SOPC board. A low-cost CMUcam color vision system is used to guide the vehicle down hallways [13]. The path to follow in the hallway of the ECE building was marked with colored poster board signs. The CMUCAM camera and processor detects and tracks color blobs. Tracking data is sent to the FPGA-based processor over a serial port.

A program (written in C) running on the processor reads the tracking data and determines how to control the speed and steer the vehicle. Like most R/C models, several pulsewidth-modulated (PWM) servo signals control the speed and steering.

After examining the hardware/software tradeoffs, students on this team decided to build PWM controllers in hardware with additional FPGA logic rather than having several complex, software, interrupt-driven timer routines running on the processor to generate the needed PWM signals. The processor simply writes the pulsewidth value to an I/O register. VHDL-based PWM state machine controllers constantly read the I/O register and generate the appropriate PWM timing signals for each of the servos. Such hardware/software tradeoffs would have been more difficult when using a traditional microcontroller-based approach.

Fig. 6 shows another interesting student design based on a small commercial robot, Amigobot [14]. This remote-controlled robot has

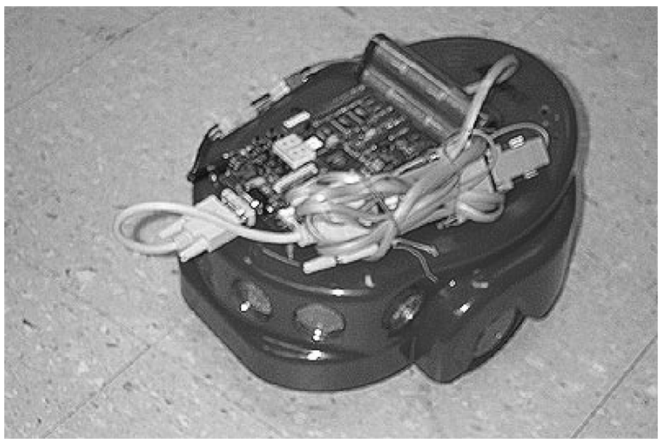

Fig. 6. Student project using the small Amigobot commercial robot controlled by the Altera Nios SOPC board.

been used in the well-known robot soccer contests. It has eight SONAR sensors, an audio system with prerecorded sounds, and two drive motors with positional feedback. A complex serial communications protocol is used to send motor commands and transmit sensor information from a microcontroller inside the robot to a remote PC via a serial port.

The Amigobot was given autonomy by replacing the PC-based remote-control function with an FPGA-based SOPC board that was mounted on top of the robot. Additional logic in the FPGA was used to add a serial port to communicate with the robot's microcontroller. A program (written in C) on the SOPC board inputs sensor data makes high-level decisions and sends motor commands using the existing serial link to the robot's internal microcontroller.

B. Using SOPC for Special Projects

The SOPC boards have also been made available for students working in other senior-level project courses. One student group designed an add-on memory board for the Digilent Digilab 2E board. Since the Xilinx Spartan-IIE FPGA on this board has a very limited amount of on-board memory, most projects utilizing the MicroBlaze soft-processor core require the additional memory for instruction and data storage. The memory add-on boards contain 256 kB of memory with a 16-b-wide output instruction/data bus. As shown in Fig. 7, the students use two memory boards in parallel to provide 512 kB of memory with a 32-b-wide instruction/data bus to the MicroBlaze soft-processor core.

V. Conclusion

Overall, using FPGA-based SOPC boards for student design projects has been a very positive development for student design projects. Using reconfigurable devices results in a higher level of abstraction over traditional design projects, thus allowing the complexity of design projects to increase. In addition, using general-purpose SOPC boards saves both time and money by minimizing the necessity for supplementary project-specific hardware that is often needed to construct prototypes. These boards have been successfully reused for several semesters on vastly different projects.

Special educational pricing for schools is available through the major FPGA vendors' university programs on the processor cores, boards, and CAD tools. This special pricing helps make SOPC an extremely attractive alternative for schools. With the educational discounts, pricing is comparable to an off-the-shelf microcontroller board.

To implement SOPC systems, students must design both the hardware and software components, requiring students to make a number of different decisions in partitioning their systems and allowing them to explore a wide range of hardware/software tradeoffs. While the complexity of these systems and the additional tools involved in the CAD tool flow for SOPC designs do present a significant learning curve for the students to overcome, difficulties can be mitigated through the use of tutorials, the enforcement of relevant prequisities (previous experience with VHDL, exposure to FPGAs, etc.), and the availability of an experienced professor or teaching assistant. Experience has proven that the projects and learning that result from SOPC design experiences are well worth the time and effort spent overcoming any obstacles.

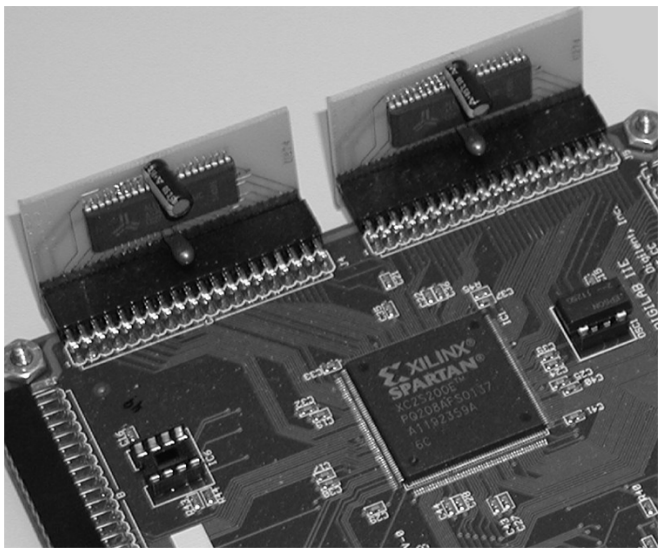

Fig. 7. Group of students designed an add-on memory board to expand the memory available to the MicroBlaze soft-processor core. In this project, two memory boards are used in parallel to provide 512 kB of memory via a 32-b-wide data bus.

Acknowledgment

A number of students and research assistants have contributed to this work during the past two years. In particular, the authors would like to thank D. Allred and A. Majid for their help in developing the memory boards and testing the Xilinx tools, as well as J. Hanson, T. Mossadak, and M. Phipps at Altera and P. Kane, D. Loftus, and A. Acevedo at Xilinx for providing software, hardware, helpful advice, and encouragement.

References

[1] C. Snyder. (2000) "FPGA Processor Cores Get Serious," Cahners Microprocessor Report. [Online]. Available: http://www.MPRonline.com/
[2] D. Seguine, "Just add sensor—Integrating analog and digital signal conditioning in a programmable system on chip," *Proc. IEEE Sensors*, vol. 1, pp. 665–668, June 2002.
[3] M. Mar, B. Sullam, and E. Blom, "An architecture for a configurable mixed-signal device," *IEEE J. Solid-State Circuits*, vol. 38, pp. 565–568, Mar. 2003.
[4] H. Chang *et al.*, *Surviving the SOC Revolution a Guide to Platform-Based Design*. Norwell, MA: Kluwer, 1999.
[5] (2002) Nios Embedded Processor User's Guide PDF File. Altera Corporation, San Jose, CA. [Online]. Available: http://www.altera.com/products/devices/nios/
[6] (2002) MicroBlaze Hardware Reference Guide, PDF Filee. Xilinx Corporation, San Jose, CA. [Online]. Available: http://www.xilinx.com/ipcenter/processsor_central/microblaze/
[7] (2002) Digilab 2E Reference Manual PDF File . Digilent, Inc., Pullman, WA. [Online]. Available: http://www.digilentine.com/Reference/
[8] J. O. Hamblen, "Using an FPGA-based SOC approach for senior design projects," in *Proc. Int. Conf. Microelectronic Systems Education*, 2003, pp. 18–19.

[9] ——, "Rapid prototyping using field-programmable logic devices," *IEEE Micro*, vol. 20, pp. 29–37, May/June 2000.

[10] J. O. Hamblen and M. D. Furman, *Rapid Prototyping of Digital Systems*. Norwell, MA: Kluwer, 1999.

[11] K. Newman, J. O. Hamblen, and T. S. Hall, "An introductory digital design course using a low-cost autonomous robot," *IEEE Trans. Educ.*, vol. 45, pp. 289–296, Aug. 2002.

[12] (2003) SOPC Projects HTML File. Georgia Institute of Technology, Atlanta, GA. [Online]. Available: http://www.ece.gatech.edu/hamblen/4006/projects/SoPC/

[13] A. Rowe, C. Rosenberg, and I. Nourbakhsh, "A low cost embedded color vision system," presented at the 2002 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Switzerland, Oct. 2002.

[14] (2001) Amigobot User's Guide PDF File. ActivMedia Robotics LLC, Amherst, NH. [Online]. Available: http://www.amigobot.com