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

Tyson S. Hall, *Student Member, IEEE,* and James O. Hamblen, *Senior Member, IEEE*
Georgia Institute of Technology, Atlanta, GA 30332–0250, hamblen@ece.gatech.edu

***Abstract*— This paper describes our experiences using a system-on-a-programmable-chip (SOPC) approach to support the development of design projects for undergraduate students in our electrical and computer engineering curriculum. A commercial field–programmable gate array (FPGA)–based SOPC development board with a reduced instruction set computer (RISC) processor core is used to support a wide variety of student design projects. A top–down rapid prototyping approach with commercial FPGA computer–aided design (CAD) 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*— system-on-a-chip, SOC, system-on-a-programmable-chip, SOPC, Field–Programmable Gate Array, FPGA, processor core, Nios, MicroBlaze, Altera, Xilinx**

I. Introduction

A new technology has emerged that enables designers to utilize a large FPGA that contains both memory and logic elements along with a intellectual property (IP) processor core to implement a computer along with custom hardware for system-on-a-chip (SOC) applications. This new approach has been termed system–on–a–programmable–chip (SOPC). In the past two years, several commercial RISC processor cores have been introduced [1]. In this paper, we 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 some example 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 ASIC and an FPGA, and 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^{TM}device. Xilinx's Virtex-II Pro family of FPGAs include up to four PowerPC processor cores on–chip. Cyress Semiconductors also offers a variation of the SOPC system. Cypress's Programmable-System-on-a-Chip (PSoC^{TM}) 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 UARTs [2], [3].

TABLE I
Features of Commercial Soft Processor Cores

Feature	Nios 3.0	MicroBlaze 3.2
Datapath	16 or 32 bits	32 bits
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 (window size: 32)	32 general purpose and 32 special purpose
Instruction Word	16 bits	32 bits
Instruction Cache	Optional	Optional
Hardware Multiplier	Optional	Optional

[1]This speed is not achievable on all devices. Some devices limit the maximum frequency to as low as 50 MHz.

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 flexibile, often allowing the designer to specify the datapath width, the 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.

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. ASICs and custom VLSI devices still offer higher performance, but they also have large development costs and longer turnaround times. 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 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 I/O hardware. This tool outputs an HDL synthesis model of the processor core in 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 GNU compiler provided with the processor core tools.

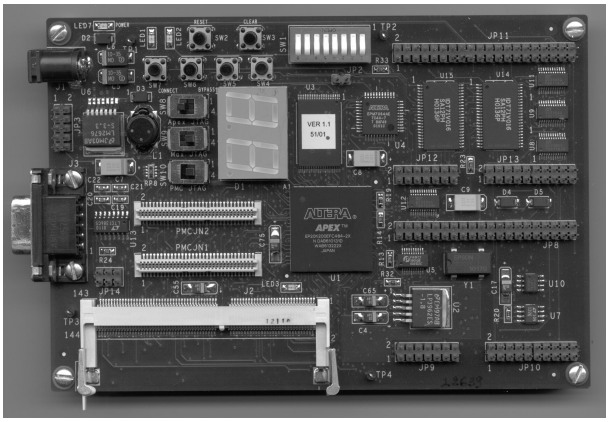

Fig. 1. Altera's Nios board contains a 200,000 gate FPGA, Flash, SRAM, several I/O options, and a RISC soft processor core for the FPGA.

B. SOPC Development Hardware

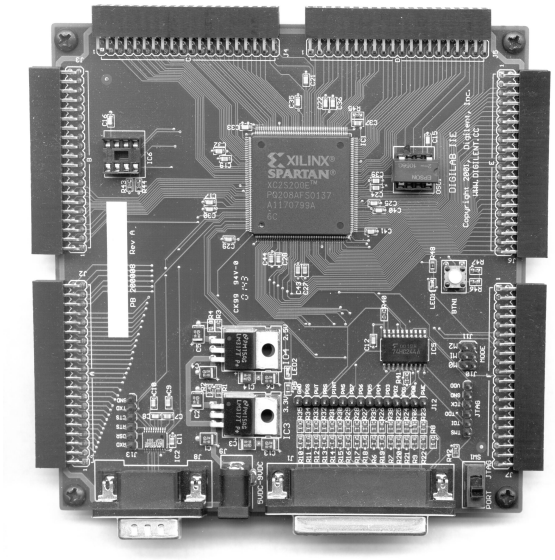

Fig. 2. Digilent's Digilab 2E low–cost FPGA board contains a 200,000 gate Xilinx FPGA that can support Xilinx's Microblaze soft core.

SOPC boards along with the required CAD tools are available from both Altera and Xilinx [4], [5]. The Altera NIOS development board shown in Figure 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 daughter board kit is available. A custom CompactFlash board can also be added if additional and/or removeable 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).

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 μClinux^{TM}kernel that runs on the board, but licensing fees still make this add–on kit somewhat expensive for student projects.

Figure 2 shows a low–cost Digilent 2E FPGA board that contains a 200,000 gate Xilinx Spartan–IIE FPGA [6]. 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 1/5th the cost of a Nios board) and can be an attractive option for student projects. To support our 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. Additionally, 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. SOC Design Using CAD Tools and FPGAs

A. Traditional Tool Flow

The traditional flow of commercial CAD tools typically follows a path from 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 step the designer through this process. As shown in Figure 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 is 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, there are a number of pre-defined processors cores available from various sources. GPL–licensed public 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.

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). Additionally, 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).

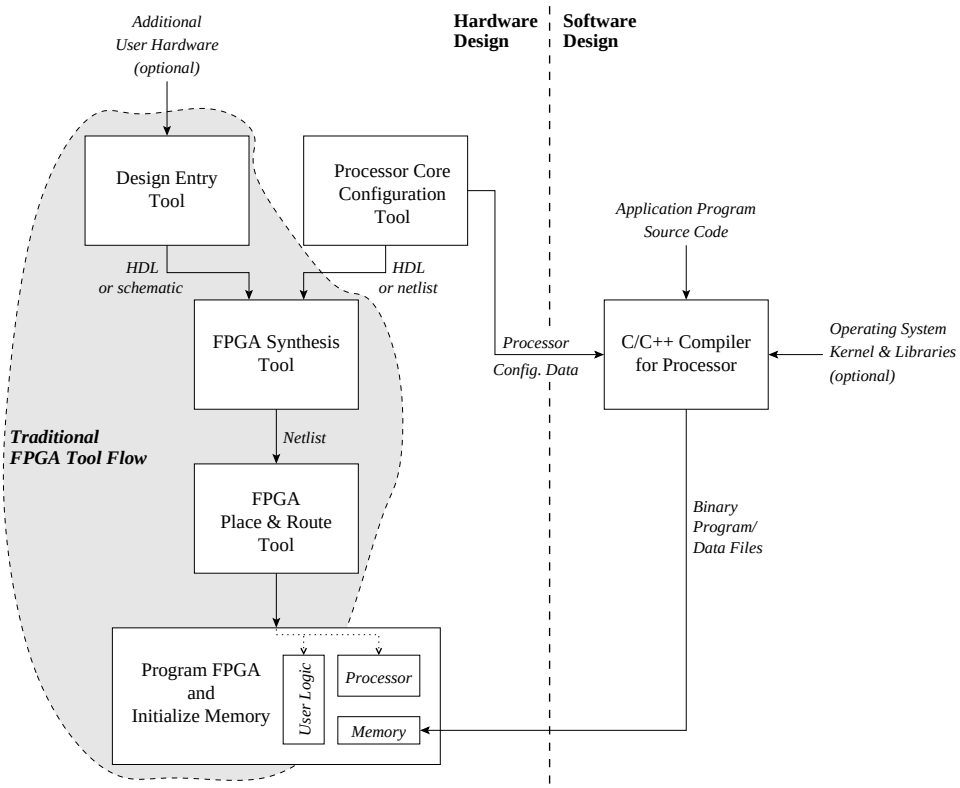

Fig. 3. The CAD tool flow for SOPC design is comprised of the traditional design process for FPGA–based systems with the addition of the Processor Core Configuration Tool and software design tools. In this figure, the program and data memory is assumed to be on–chip for simplicity. Figure 4 shows a more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

In the case of Altera and Xilinx, the Processor Core Configuration Tool block shown in Fig. 3 is realized in a user–friendly GUI interface that allows the designer to customize the processor for a particular project. The configurable parameters include the datapath width, memory, address space, and peripherals (including arbitrarily defined general–purpose I/O, UARTs, Ethernet controllers, memory controllers, etc.). 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. The next step is to write and compile the software that will be executed on the soft processor core.

C. High–level Compiler for Processor Core

When the Procesoor 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 standalone programs to run on the processor. Optionally, the designer can compile code for an operating system targeted for the processor core. Altera sells an add–on kit that includes a version of μClinux^{TM}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 can be done several ways depending on the memory configuration of the processor at hand.

a) 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 Sect. III-A). This 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 is not an option.

b) 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 their 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 to 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.

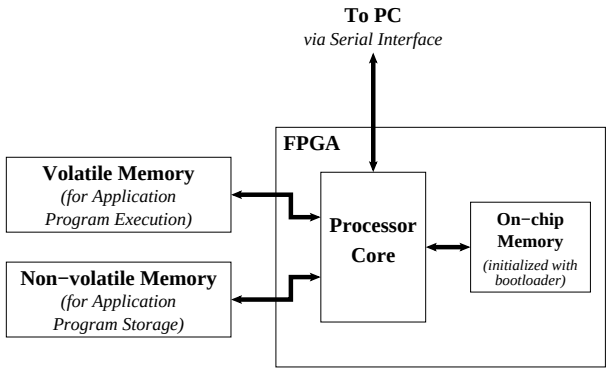

Fig. 4. This arrangement of on–chip and external memories provides flexiblity and performance to an SOPC system. The internal memory stores a bootloader program that can retrieve an application program from the non-volatile memory or the PC via the serial interface and store it in volatile memory for execution. Additionally,the non-volatile 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 (non-volatile memory); and flexibility in programming is achieved through the bootloader and serial interface.

c) External Nonvolatile Storage: The application program code can be stored on an external EEPROM, Flash, or other form of non-volatile memory. The application program can either be pre-programmed 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 program between the memories. The fast, volatile memory (i.e., SDRAM) is used to store the application program during execution. Finally, the slower, non-volatile memory (i.e., Flash or EEPROM) is used for the permanent storage of the application program. The bootloader program can be modified to, on power–up, retreive a program from non-volatile 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 Cirriculum

For the past four semesters, we 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.

Based on our experience with the existing SOPC tools, students need to have taken previous coursework in digital logic design, computer architecture, and C progrmming [7], [8], [9]. Some prior experience in VHDL or Verilog and exposure to FPGAs and their associated CAD tools is also useful. In most undergraduate curricula, this 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 it is 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, we now require students to successfully complete a system–level tutorial and demo using a SOPC reference design during the first few weeks of any project course. This forces them to start work earlier and to already be familiar with the SOPC boards hardware and the complex CAD tool flow before the project specific work starts.

We have found that it is still necessary to have an experienced user such as a course instructor or teaching assistant to install and maintain all of the CAD tools and to be available to help students when they occasionally encounter hardware and tool–related problems that they cannot resolve.

A. Using SOPC in Senior Design

ECE 4006, Major Design Project, is our 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 trade–offs. A number of the student teams have used the SOPC approach to construct a prototype of their design. Projects have included web servers, email servers, vision systems, Internet appliances, and numerous robots. 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. A student project using a small hobbyist R/C Hummer Vehicle with the color tracking CMUCAM all controlled by the Altera NIOS SOC board.

Figure 5 shows a student robotics project. An off-the-shelf hobbyist radio–controlled 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 [10]. 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 C program 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 pulse-width 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 pulse width 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.

Figure 6 shows another interesting student design based on a small commercial robot, Amigobot [11]. This remote controlled robot has been used in the well–known robot soccer contests. It has eight SONAR sensors, an audio system with pre-recorded 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 C program 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.

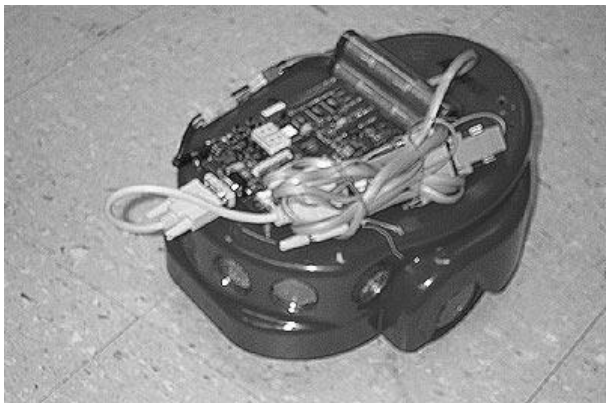

Fig. 6. A student project using the small Amigobot commercial robot controlled by the Altera Nios SOPC board.

B. Using SOPC for Special Projects

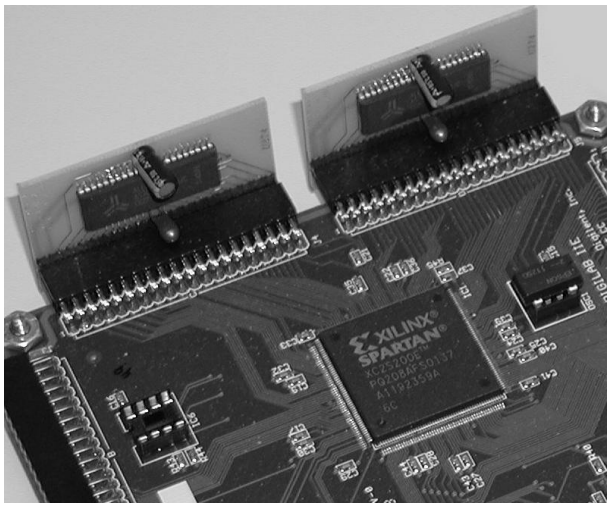

Fig. 7. A group of students designed an add–on memory board to expand the memory available to the MicroBlaze soft processor core. In this project, two memory boards are used in parallel to provide 512 KB of memory via a 32–bit wide data bus.

We have also made the SOPC boards 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. 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 required the additional memory for instruction and data storage. The memory add–on boards contain 256 KB of memory with a 16–bit 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–bit wide instruction/data bus to the MicroBlaze soft processor core.

V. Conclusions

Overall, using FPGA–based SOPC boards for student design projects has been a very positive development for our student design projects. The complexity of the design projects has increased since introduction of the boards. Having a general–purpose SOPC board saves both time and money, because students don't have to order and wait on as many parts for design projects. The amount of additional hardware needed for construction of the prototype is greatly reduced, and our 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 helps make SOPC an extremely attractive alternative for schools. With the educational discounts, pricing is comparable to an off–the–shelf microcontroller board.

The additional tools involved in the CAD tool flow for SOPC designs do present a significant learning curve for the sutdents to overcome. However, difficulties working with the development software and SOPC boards 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. It has been our experience that the projects and learning that result from SOPC design experiences are well worth the time and effort spent overcoming any obstacles.

VI. Acknowledgements

A number of students and research assistants have contributed to this work during the past two years. In particular, we would like to acknowledge Daniel Allred and Ashrif Majid for their help in developing the memory boards and testing the Xilinx tools. Joe Hanson, Tawfiq Mossadak, and Mike Phipps at Altera and Patrick Kane, David Loftus, and Anna Acevedo at Xilinx provided software, hardware, helpful advice, and encouragement.

Tyson S. Hall is a PhD student in electrical and computer engineering at Georgia Tech. His current research interests include rapid prototyping of mixed signal systems, cooperative analog/digital signal processing, reconfigurable computing, and embedded systems. Hall received an MSECE from Georgia Tech in '01 and a BSCMPE from Georgia Tech in '99.

James O. Hamblen is a professor in electrical and computer engineering at Georgia Tech. His current research interests include rapid prototyping, high–speed parallel and VLSI computer architectures, computer–aided design, and reconfigurable computing. He received a PhD in Electrical Engineering from Georgia Tech, an MSEE from Purdue University, and a BEE from Georgia Tech. Prior to earning his PhD, he worked as a systems analyst for Texas Instruments in Austin, Texas, and as a senior engineer for Martin Marietta in Denver, Colorado.

References

[1] C. Snyder, "Fpga processor cores get serious," in *Cahners Microprocessor Report*, http://www.MPRonline.com/, Sept. 2000.

[2] D. Seguine, "Just add sensor - integrating analog and digital signal conditioning in a programmable system on chip," *Proceedings of IEEE Sensors*, vol. 1, pp. 665–668, 2002.

[3] M. Mar, B. Sullam, and E. Blom, "An architecture for a configurable mixed-signal device," *IEEE Journal of Solid–State Circuits*, vol. 38, no. 3, pp. 565–568, Mar. 2003.

[4] *Nios Embedded Processor User's Guide*, PDF File, Altera Corporation, http://www.altera.com/products/devices/nios/, Jan. 2002.

[5] *MicroBlaze Hardware Reference Guide*, PDF File, Xilinx Corporation, http://www.xilinx.com/ipcenter/processor_central/microblaze/, Apr. 2002.

[6] *Digilab 2E Reference Manual*, PDF File, Digilent, Inc., http://www.digilentinc.com/Reference/, Apr. 2002.

[7] J. O. Hamblen, "Rapid prototyping using field–programmable logic devices," *IEEE Micro*, vol. 20, no. 3, pp. 29–37, May/June 2000.

[8] J. O. Hamblen and M. D. Furman, *Rapid Prototyping of Digital Systems*. Kluwer Academic Publishers, 1999.

[9] K. Newman, J. O. Hamblen, and T. S. Hall, "An introductory digital design course using a low–cost autonomous robot," *IEEE Transactions on Education*, vol. 45, no. 3, pp. 289–296, Aug. 2002.

[10] A. Rowe, C. Rosenberg, and I. Nourbakhsh, "A low cost embedded color vision system," in *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Switzerland, Oct. 2002.

[11] *Amigobot User's Guide*, PDF File, ActivMedia Robotics LLC, http://www.amigobot.com, 2001.