An Evolving Curriculum to Match the Evolution of Reconfigurable Computing Platforms

Graham Schelle*, Daniel Fay†, Dirk Grunwald*, Dan Connors† and John Bennett†
*Dept. of Computer Science
University of Colorado at Boulder
Boulder, CO USA
†Dept. of Electrical and Computer Engineering
University of Colorado at Boulder
Boulder, CO USA
{schelleg,faydr,grunwald,dconnors,jkb}@colorado.edu

***Abstract*— Reconfigurable platforms have evolved from "sea of gates" architectures into diverse System on a Chip (SoC) platforms with embedded processor cores and dedicated hardware components. This evolution has greatly increased the performance of this technology, but creates challenges when teaching the new technology to Computer Science and Electrical Engineering graduate students. Previously, knowledge of Hardware Description Languages (HDLs) was the only prerequisite for advanced courses in reconfigurable computing, but now knowledge of embedded processors, complex tool suites, and hardware/software co-design form the basic foundation of education in this field. At the University of Colorado at Boulder we have partnered with Xilinx, a leading manufacturer of FPGAs, and endeavored to create a curriculum that addresses these challenges. Through a graduate level embedded systems course that hybridizes microprocessor-based software with dedicated hardware and various seminar courses on reconfigurable computing, we have been able to observe how students best learn subject matter in this area and what subject matter they find challenging. In this paper, we describe our experience teaching reconfigurable computing with an emphasis on programmable SoC design.**

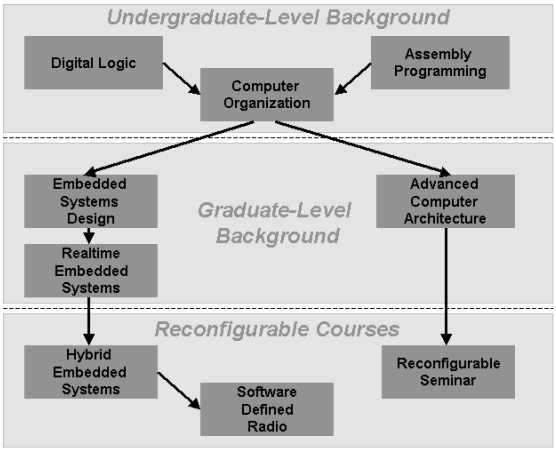

Fig. 1. Reconfigurable course sequence at the University of Colorado.

I. Introduction

As the diversity and complexity of reconfigurable computing increases, so must the university curriculum that teaches it. Cutting-edge reconfigurable computing education not only requires knowledge of digital design, but also a deep understanding of hardware/software codesign. In addition to understanding logic and embedded microprocessors, students need to know how to divide an application between logic and processors and how to select the correct tools for design and debug. As reconfigurable computing evolves toward System on a Chip (SoC) architectures, reconfigurable computing education must follow [1].

Modern FPGA designs reflect the move toward SoC architectures for reconfigurable computing. Originally a "sea of gates" used mainly as glue logic, FPGAs have evolved to become powerful compute engines containing specialized hardware blocks like fast carry chains, embedded multipliers, SRAMs, and full-fledged microprocessor cores.

In parallel to the increased use of embedded components, FPGA designs that use soft IP (Intellectual Property) blocks are now commonplace. The standard soft IP includes, at a minimum, microprocessors, I/O controllers, and various kinds of memories. Any FPGA-based reconfigurable computing course that ignores these embedded and soft components will not allow students to realize the full potential of modern FPGAs. As a result, a reconfigurable course must emphasize this block-based design philosophy over the traditional, gate-based logic design methodology, relegating digital logic courses to teaching the latter.

Embracing these trends, our reconfigurable curriculum now utilizes a programmable System on a Chip design flow. Programmable System on a Chip refers to the various execution configurations that can exist on modern FPGAs, which contain a mixture of components (embedded and soft processors, IP components) linked together by a variety of interconnects (buses, Network on Chip, specialized). This design methodology differs starkly from traditional HDL-based FPGA designs, as it requires a holistic approach that incorporates both hardware and software into the overall system design.

Figure 1 shows our graduate reconfigurable curriculum, which consists of a course in hybrid embedded systems,

a reconfigurable seminar course, and a course in Software Defined Radio (SDR). Our hybrid embedded systems course provides students a foundation for further SoC studies by teaching them how to use the Xilinx tools to realize the full capabilities of modern FPGAs. Our seminar course provides an overview of current reconfigurable computing research. Finally, our SDR course uses the powerful DSP capabilities of contemporary FPGAs to digitally modulate and demodulate analog radio signals.

The rest of this paper discusses both how we developed our reconfigurable computing curriculum as well as the challenges we faced with these new classes. Section II discusses the infrastructure improvements and partnerships with industry critical to our new curriculum's success. Section III discusses our experiences developing the Hybrid Embedded Systems course, sections IV and V present the relevant seminar courses, and we conclude the paper by discussing the future directions of our curriculum.

II. Infrastructure Development

In this section, we discuss the hardware and software foundations that allow our curriculum to keep current with new developments in reconfigurable computing.

Hardware Platforms. The hardware available to the students has greatly increased due to our partnership with Xilinx. Through various research projects, courses, and individual contacts, we have procured a variety of FPGA boards. The Software Defined Radio course, for example, uses the Nallatech XtremeDSP boards for signal processing applications, while the Hybrid Embedded Systems class uses the Digilent XUPV2P board.

Software Tools. Modern reconfigurable computing architectures use multiple layers of tools, scripts, and GUIs for development. This software rises above standard HDL design, augmenting it with higher-level schematic- and script-based design flows. This allows students to focus on larger system design issues and quickly push complicated designs into hardware. Hybrid Embedded Systems leverages Xilinx's Platform Studio to facilitate integrating software and hardware into a single working design, while the Matlab / Simulink / System Generator for DSP tool chain allows students to quickly reify their Matlab designs into working hardware implementations. While different classes require different FPGA hardware platforms, they all use the same suite of Xilinx software tools, which means students need only learn the tools once.

Technical Support. Historically, FPGA design tools required dedicated staff to maintain servers and specialized workstations, apply updates, and answer student inquiries. In the last few years, however, the licensing and online support have improved to where the students can find the technical support they need on their own. Additionally, the tools themselves can now be installed on commodity Windows-based laptops without needing to access a license server. This is in part due to Xilinx providing a great deal of free software for the students as well as other EDA vendors providing their tools at significantly reduced cost.

III. Hybrid Embedded Systems

A. Goals

Historically, embedded systems have been designed as primarily either software-centric or hardware-centric designs, where the design effort centers around either writing software for a microprocessor/microcontroller or making a specialized hardware design.

A new course we developed, called Hybrid Embedded Systems [2], sets out to bridge the gap between these two design methodologies in order to gain the benefits of both designs. At the University of Colorado, it replaces the Embedded Systems Lab course as the third course in the three-course sequence known as the *Embedded Systems Certificate* courses in the Electrical and Computer Engineering department. The other two classes in this sequence are Embedded Systems Design [3], a class based around the 8051 microcontroller, and Realtime Embedded Systems [4], which teaches students about realtime systems using the Wind River VxWorks [5] operating system running on Intel Pentium-based hardware systems. Both of these classes teach embedded systems from a processor-centric perspective – most of the design work involves programming the microprocessor such that it acts as a mediator between the other components in the system.

While there are other hybrid hardware/software design methodologies, teaching the SoC design flow is ideal for a semester-long course. In addition to learning about a widely-used, cutting-edge, design technique, SoC design allows students to explore a wide range of different hardware/software partitions without spending large amounts of time writing, synthesizing, debugging, and testing custom logic. Instead, students can evaluate different ways to partition the system between hardware and software simply by connecting different pre-existing hardware blocks together using a set of standardized interconnection schemes.

B. Curriculum

Figure 2 shows the three boards that have been used in Hybrid Embedded Systems: on the left is the Insight Memec DS-KIT-2VP4LC-NE board, in the center is the Digilent XUPV2P board, and on the right is the Xilinx ML310 board. Table I provides a comparison of the features offered by the three boards. Hybrid Embedded Systems' debut semester used primarily the Insight Memec board, although a few final projects used the ML310.

As the first semester of Hybrid Embedded Systems went on, the Insight Memec board became inadequate for the following reasons:

1) **Insufficient logic resources.** Creating a substantial SoC design requires large amounts of logic to instantiate all of the hardware blocks and interconnection bus logic.
2) **Lack of onboard memory.** The Insight Memec board does not come with any external SRAMs or DRAMs, which means that programs must fit into the few kilobytes of Block RAMs provided by its small FPGA. There is not enough memory to boot, for example,

Feature	Memec	ML310	XUPV2P
Logic Resources (slices))	6,768	30,816	30,816
On-Chip Memory (kbits)	504	2448	2448
PowerPC 405s	1	2	2
Audio CODEC	No	Yes	Yes
Video RAMDAC	No	No	Yes
LCD Display	Yes	Yes	No
External Memory	No	DDR DIMM	DDR DIMM
RS-232 Port	Yes	Yes	Yes
Built-in Speaker	Yes	No	No
Switches/Pushbuttons	Yes	Yes	Yes
LEDs	Yes	Yes	Yes
GPIOs	Yes	Yes	Yes
CompactFlash Interface	No	Yes	Yes
Built in JTAG Controller	No	No	Yes
Ethernet	No	No	Yes
Hard Drive Interface	No	IDE	SATA
PCI Slots	No	Yes	No

TABLE I

FEATURE COMPARISON OF THE INSIGHT MEMEC DS-KIT-2VP4LC-NE[6], XILINX ML310[7], AND THE DIGILENT XUPV2P[8].

Embedded Linux or to create large, data-intensive applications.

3) **Lack of external peripherals.** A major use of SoCs is to quickly process large amounts of data in real time. The Insight Memec board does not have any video input, video output, or sound input facilities. Additionally, there is no way to interface the board with mass storage devices such as hard drives or flash devices.

The next semester, in the Fall of 2005, the course switched to using a new board, the Digilent XUPV2P board. Compared to the Insight Memec board, the XUPV2P contains a larger FPGA, access to external memory, as well as having a large number of input/output devices directly accessible from the FPGA, such as a video RAMDAC, an AC97 audio CODEC, Ethernet, Serial ATA hard drive connectors, a CompactFlash interface, and more general-purpose I/O (GPIO) connectors. This is in contrast to the ML310, whose similar featureset is accessed through the PCI bus, complicating projects that do not use an embedded operating system like Embedded Linux or VxWorks. While many of these features can be obtained for the Insight Memec board, they require the purchase of additional hardware.

Another advantage that the XUPV2P board has over the Insight Memec board is that it is a significantly better value: for academic customers, the XUPV2P board costs US$299, while the Insight Memec board costs US$199 and the ML310 costs US$999. The cost advantage that the Insight Memec has over the XUPV2P board becomes smaller when one takes into account the XUPV2P's built-in USB-based JTAG connectivity.

The class itself is divided into two parts. The first half of the class consists of a lecture discussing theory as well as the operation of the tools and equipment used during the semester. The lecture consisted of the following parts:

1) **VHDL. (3 weeks)** Provide an introduction to VHDL, which is the "supported" HDL for this course. Also part of the VHDL teaching is how VHDL is used for synthesis and simulation. This gives some insight into how simulators work.
2) **Basic hardware design using the Xilinx Integrated Software Environment (ISE) (1 week).** Show the students how to do basic logic design using the Xilinx ISE tools, as well as how to use the tools to synthesize the design.
3) **Introduction to the Embedded Developers' Kit (EDK) (3 weeks).** Students learn to add and configure IP blocks, including custom IP that they develop.
4) **Debugging Tools (1/2 week).** Students learn how to use ChipScope Pro, GDB, and XMD.
5) **Hardware/Software co-design (1/2 week).**
6) **Profiling (1 week).** Students learn to evaluate software performance, and to develop hardware accelerators for critical functions.
7) **Floating-point (1/2 week).** Students learn the basics of adding floating-point functionality to their designs.
8) **Multiple processors (1/2 week).** Students learn the basics of multiprocessing, including the implementation of multiple processor designs.

In addition to the lecture material, there were also structured laboratory assignments. Assigned every other week, the students receive a new assignment that serves as a tutorial to teach the tools used in the class.

1) **Lab Familiarization; VHDL Tutorial.** This is the standard introductory lab, where the goal is to just make sure that everything works for the student. This lab also introduced the students to VHDL.
2) **ISE/ModelSim/VHDL.** A tutorial that introduces the students to developing logic designs in Xilinx ISE and simulating those logic designs using ModelSim SE.
3) **ISE/VHDL.** In this lab, the students made and implemented a simple 4-bit binary counter on the XUPV2P board so that they became familiar with synthesizing simple logic designs to hardware.
4) **EDK Base System Builder (BSB).** This lab introduced the students to Base System Builder, a wizard-based tool for creating working hardware/software platforms to be used as a foundation for more advanced designs.
5) **Using the EDK.** Continues where the previous lab left off, by introducing the students to adding IP blocks to the design made in Base System Builder.
6) **SW Development and Debugging.** Introduces the debugging tools XMD, GDB, and ChipScope Pro.
7) **Advanced Topics.** This final lab gives the students hands-on experience with performance tuning. Experience with profiling, caching, and hardware acceleration.

C. Final Projects

In the two semesters that this class has been offered, a variety of final projects have been completed by students. Some of the best of these are described below:

- **Parameterizable cache emulator** This project conducted microarchitectural design exploration by using the FPGA as a parameterizable hardware cache emulator.

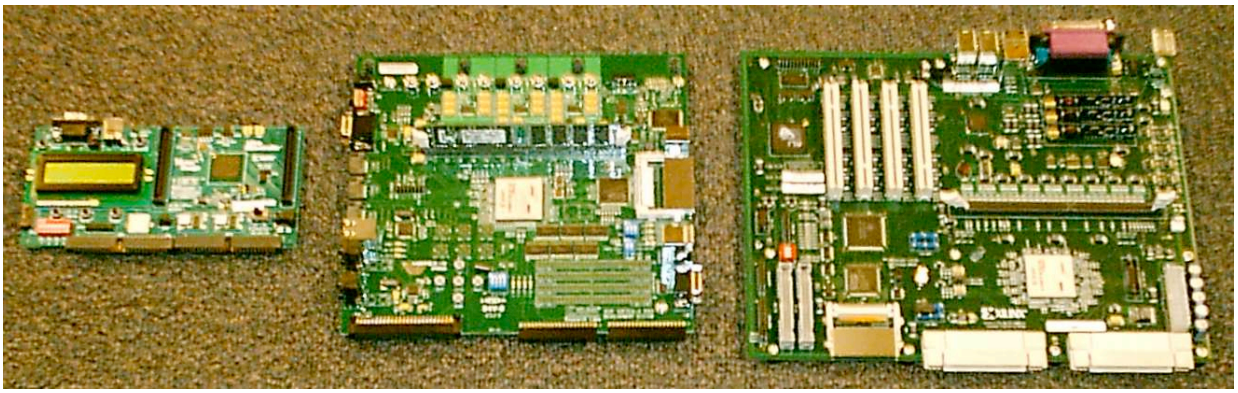

Fig. 2. Boards used in Hybrid Embedded Systems: Insight Memec DS-KIT-2VP4LC-NE (left), Digilent XUPV2P (center), and Xilinx ML310 (right).

- **MP3 player using APU acceleration** The students doing this project used the Auxiliary Processing Unit (APU) interface on the PowerPC 405 contained on a Virtex-4 to interface with hardware accelerating MP3 playback.
- **Hardware Edge Detection** This project implemented an edge detection algorithm in hardware to detect shapes in real-time video. Input is through the video card, output is to a TV screen, and information is displayed.
- **Audio Streaming over Ethernet** This group created a real-time MP3 streaming server. The overall functionality is where one can connect an audio source to the audio-in jack of the XUPV2P board, and have the input encoded into an MP3 and sent out over Ethernet to the network as either an HTTP stream or Icecast/Shoutcast stream. The design used two PPC405s to move data from the audio CODEC, to the hardware MP3 accelerator, and out through the Ethernet port.
- **AC-3 Audio encoding/Decoding** This group developed and AC3 audio encoder/decoder. Since the computation requires a significant amount of floating-point computation, the group integrated a PLB-attached floating-point unit into their design.

IV. Reconfigurable Computing Seminar

In addition to the Hybrid Embedded Systems course, we have offered a seminar course to our graduate students on reconfigurable computing. This course consists of reading 1-2 papers a class and having group discussions culminating in group research projects. Having these classroom discussions allows instructors to examine the diverse body of work that encompasses reconfigurable computing and determine what is most interesting in terms of education and research.

Interestingly, the term reconfigurable computing has been increasingly used to describe a variety of platforms, not just FPGAs. With the DARPA Polymorphic Computing Architectures Funding in 2000, many projects arose that raised the level of reconfigurability to tile-based multiprocessor designs connected by a Network on Chip (NoC) [9] [10]. It is important to include these projects when discussing reconfigurable computing, as traditional FPGAs have also left the pure logic arena embedding processors, high speed transceivers, and DSP cores. This observation has led us to gear our education in the seminar courses towards the entire spectrum of reconfigurable computing, beginning with fine-grained FPGA architectures and ending with multicore processor designs. Simply discussing FPGA architectures and applications is no longer enough when discussing reconfigurable computing.

The seminar course is available to both ECE and CS graduate students. As with the hybrid embedded systems course, the students enter with a variety of background knowledge and experience. However, in our experience, most students are typically interested in systems research (e.g. architecture, compilers, networks). Since this class consists mainly of reviewing and discussing papers, students from a variety of backgrounds can participate. In the future, we expect that students from other areas such as parallel processing will use this seminar course to complement to their research in computer science and engineering.

A. Topics Of Interest

The reconfigurable computing seminar covers roughly 50 papers over the course of a semester. We do not enumerate the papers here, but categorize areas of interest. Overall, the goal is to present a variety of architectures, applications, and tools that are directly related to reconfigurable computing.

- **Coarse Grained Reconfigurable Architectures.** We begin the class by exploring the variety of reconfigurable computing architectures that exist. GARP [11] and RaPid [12] are two examples of these architectures. These non-traditional architectures (i.e. not solely FPGA logic) present how heterogeneous components can work together in reconfigurable computers. We emphasize the aspects of performance and usability to the software and hardware designer communities. The lack of accepted benchmarks for these designs has often frustrated students when making performance comparisons. However, the research does good comparisons against standard processor implementations of application sets.
- **Application Space Driven Architecture Exploration.** With so many different configurations and architectures

in reconfigurable computing, we then discuss architecture exploration tools [13]. This part of the class is particularly interesting in that reconfigurable computing architecture research papers typically present a given architecture and then map applications onto it. With these tools, however, one instead presents the application and then finds the best architecture for it. Students found this section helpful in comparing what architectural features are most important to a given application set.

- **Network on Chip Interconnect.** We discuss NoC research as a needed component to System on Chip designs [14]. We discuss how power consumption issues, leads many architects to move toward NoC architectures. We emphasize this in order to compare the current state of reconfigurable computing with that of well-understood processor designs. Students with a good knowledge of processor design are interested in the challenges of creating a Network on Chip that meets processor communication needs. Networking students also find this section of the class novel and interesting due to the network challenges that come from the "onchip" nature of NoCs.
- **High Level Reconfigurable Computing Languages.** Discussing the difficulties in writing "code" for reconfigurable computers is of particular interest to computer science students. We explore various languages that are used to target specific platforms, emphasizing the performance tradeoffs with writing raw VHDL and C code [15] [16]. Comparison of languages against each other make for good semester projects, exposing the students to multiple languages and mapping those languages down to implementations.

V. Software Defined Radio Seminar

We briefly present our Software Defined Radio (SDR) seminar here as an example of an emerging research field that benefits greatly from having a reconfigurable computing curriculum in place.

The course itself is not focused on reconfigurable computing; rather, it is more concentrated in DSP concepts and the software tools that exist to create and simulate SDR algorithms. Topics include smart antennas, signal filtering, and example SDR platforms. SDR is an excellent application space where reconfigurable computing platforms can allow students to implement their designs. Specifically, using Nallatech's XtremeDSP boards along with the Matlab/Simulink/System Generator for DSP tool flow, students can place SDR algorithms onto real hardware. Without other reconfigurable computing courses, especially the hybrid embedded systems course, we would be unable to teach students how to use the reconfigurable computing platforms within a semester. Currently, students unfamiliar with the Xilinx tool flow can use alternate implementation platforms, notably the GNU Software Radio project [17].

VI. Future of Reconfigurable Computing Education

Reconfigurable Computing education at the University of Colorado has grown from simple logic design into complex SoC design education. In the future, the curriculum will need refinement in tool selection, and an emphasis on reconfigurable computing in other research fields. We also expect that the Hybrid Embedded Systems course will continue to evolve to match the increasing complexity of designs.

A. Tool Selection Challenges

There are a number of tools that can be used to develop SoC designs. From high level languages to the standard FPGA vendor tools, it has been difficult to determine the best ones for students to learn. In our experience, the best support and maintenance has come from established vendors (Xilinx, Mentor Graphics, etc.), but using less supported newer tools may be a benefit to students (SystemC, Confluence, etc.). We still need to reduce the number of tools used in our curriculum, and we need to integrate their usage into other courses. For the present, we are staying with the Xilinx and Matlab design flows, as they are well-supported by industry.

B. Reconfigurable Computing Integration into Other Fields

We also are making efforts to introduce reconfigurable computing into existing and new graduate-level courses. As noted in section V, we currently offer a seminar course in the area of Software Defined Radio. This experience can be applied to other seminar courses in a variety of subject areas.

For example, architecture courses could benefit from exploration tools such as SystemC, allowing students to quickly see how design choices affect a variety of performance metrics. In parallel processing courses, we can use the availability of multiple RISC processors onchip to do real implementations of concepts learned in class. Finally, in compiler courses, the increasing complexity of compilers can be made apparent by examining hybrid reconfigurable computing platforms.

C. Future Developments in the Hybrid Embedded Systems Course

In the future, less course time needs to be spent on tutorials to acquaint the students with the tools and hardware. One way to do this is to reduce the number of labs by combining several tutorial labs into one lab, since each lab takes only a few hours. Similarly, the lecture portion of the class should spend less time on tool-related topics and more time on theory.

Future offerings of this course need to provide some discussion of appropriate hardware/software tradeoffs in hybrid embedded systems. Examples include how control-intensive code is most appropriate for a microprocessor-type device, the powerful capabilities of DSP code in an FPGA, and the places where FPGAs are particularly useful, such as when the application exhibits large amounts of fine-grained parallelism and/or the data types have non-standard bitwidths. The major bottlenecks to high performance in SoC designs, such as on-chip/offchip bandwidth and latency, need to be discussed as

well. Other issues that merit discussion include time-to-market versus cost and performance tradeoffs, and concurrent designs that have multiple processing elements working at once.

Additionally, the course could provide some treatment of issues concerning power consumption, an issue that is critically important in many current embedded systems, and will become even more important in the future. Xilinx provides a tool, called XPower, that estimates the power consumed by the FPGA using ModelSim traces as input.

Since there are many possible directions the course can take, an option might be to let students guide the learning. We could allow the students, for the last few labs, to choose from several different possible design choices. Possible topics could include:

1) **DSP.** Have a lab that deals with issues specific to using the FPGAs as powerful Digital Signal Processors (DSPs).
2) **Multiprocessing.** Have a tutorial which shows the students how to create multiprocessor setups. Such setups could include making a heterogeneous system with multiple, different cores, e.g. two PowerPC processors and/or one or more MicroBlaze processors.
3) **Fault tolerance.** As the features on integrated circuits become ever smaller, their sensitivity to radiation-induced errors increases. Additionally, harsh operating environments, such as outer space, place new demands on integrated circuits. A lab that explores techniques to mitigate these problems, such as error-correcting codes (ECC) for memories and running logic in lockstep, would allow students to gain experience making fault-tolerant systems.
4) **Multi-gigabit I/O.** The Xilinx Virtex-II Pro FPGAs have several Rocket I/O multi-gigabit transceivers, which are differential serial lines running at multiple gigabaud. These lines can be used for implementing Serial ATA (SATA) disk interfaces.
5) **Embedded Operating Systems.** These could be done using either the XUPV2P or the ML310, to study FPGA-based systems running either Embedded Linux.

VII. Conclusions

We have described the evolving graduate level reconfigurable computing curriculum at the University of Colorado. As students' interests have progressed to using the variety of embedded cores on modern FPGAs, we have adapted the curriculum to teach programmable System on Chip design and research practices. Balancing training on one specific tool flow versus educating general design processes has been the most notable challenge in Hybrid Embedded Systems. The seminar course has identified interesting areas of research in reconfigurable computing, but must remain responsive to the rapidly evolving technology and the students' background knowledge and interests. We believe that these courses will increase in popularity, as have the related embedded systems courses that are currently offered.

References

[1] H. D. Man, "System-on-chip design: Impact on education and research." *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 11–19, 1999.

[2] "Ecen 5633 hybrid embedded systems," http://ece.colorado.edu/academics/courses/ECEN_5633.html.

[3] "Ecen 5613 - embedded system design," http://ece.colorado.edu/academics/courses/ECEN_5613.html.

[4] "Ecen 5623 - real-time embedded systems," http://ece.colorado.edu/academics/courses/ECEN_5623.html.

[5] "Vxworks 6.0," http://www.windriver.com/products/device_technologies/os/.

[6] "Virtex-ii pro lc development kit," http://www.memec.com/uploaded/VirtexIIProLC_1.pdf.

[7] "Ml310 documentation and tutorials," http://www.xilinx.com/products/boards/ml310/current/index.html.

[8] "Virtex-ii pro development system," http://www.digilentinc.com/info/XUPV2P.cfm.

[9] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: a computational fabric for software circuits and general-purpose programs," in *Micro, IEEE*, vol. 22. IEEE Computer Society Press, 2002, pp. 25–35.

[10] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, N. Ranganathan, D. Burger, S. W. Keckler, R. G. McDonald, and C. R. Moore, "Trips: A polymorphous architecture for exploiting ilp, tlp, and dlp," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 62–93, 2004.

[11] J. R. Hauser and J. Wawrzynek, "Garp: A MIPS processor with a reconfigurable coprocessor," in *IEEE Symposium on FPGAs for Custom Computing Machines*, K. L. Pocek and J. Arnold, Eds. Los Alamitos, CA: IEEE Computer Society Press, 1997, pp. 12–21. [Online]. Available: citeseer.ist.psu.edu/hauser97garp.html

[12] C. Ebeling, D. C. Cronquist, P. Franklin, J. Secosky, and S. G. Berg, "Mapping applications to the rapid configurable architecture," in *Proceedings of the 5th IEEE Symposium on FPGA-Based Custom Computing Machines (FCCM '97)*. IEEE Computer Society, 1997.

[13] M. Wan, H. Zhang, V. George, M. Benes, A. Abnous, V. Prabhu, and J. Rabaey, "Design methodology of a low-energy reconfigurable single-chip dsp system," *J. VLSI Signal Process. Syst.*, vol. 28, no. 1-2, pp. 47–61, 2001.

[14] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the Design Automation Conference*, Las Vegas, NV, June 2001, pp. 684–689.

[15] M. B. Gokhale, J. M. Stone, J. Arnold, and M. Kalinowski, "Stream-oriented fpga computing in the streams-c high level language," in *FCCM '00: Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*. Washington, DC, USA: IEEE Computer Society, 2000, p. 49.

[16] L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, and M. Poncino, "Systemc cosimulation and emulation of multiprocessor soc designs," *Computer*, vol. 36, no. 4, pp. 53–59, 2003.

[17] "Gnu radio - the gnu software radio," http://www.gnu.org/software/gnuradio/.