Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2014, Article ID 287205, 12 pages
http://dx.doi.org/10.1155/2014/287205

Research Article

Practical Education Fostered by Research Projects in an Embedded Systems Course

Vanderlei Bonato,[1] Marcio M. Fernandes,[2] Joao M. P. Cardoso,[3] and Eduardo Marques[1]

[1] *Department of Computer Systems, Institute of Mathematical and Computing Sciences, The University of Sao Paulo, 13560-970 São Carlos, SP, Brazil*
[2] *Department of Computer Science, Federal University of São Carlos, 13565-905 São Carlos, SP, Brazil*
[3] *Department of Informatics Engineering, Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal*

Correspondence should be addressed to Vanderlei Bonato; vbonato@icmc.usp.br

Received 30 January 2014; Accepted 21 May 2014; Published 29 June 2014

Academic Editor: Michael Hübner

The very nature of universities makes them unique environments for research and teaching. Although both activities constantly borrow from each other, a deeper level of interaction is not always achieved for several reasons. This paper presents a successful experience on conducting an undergraduate course on embedded systems, based on strong interaction with related research activities previously conducted by the authors. Known for being everywhere, embedded systems are constantly expanding in both complexity and volume production. In addition, heterogeneous systems are becoming prevalent in modern applications, standing as an additional difficulty to students in this area. In this context, this paper presents experiences in teaching embedded systems using a project-based learning pedagogical approach, with strong emphasis on mobile robotic applications previously developed by MSc and PhD students. As a result, it has been observed that undergraduate students have the opportunity to build a strong background and feel better prepared to face the challenges to be found in their future professional activities.

1. Introduction

In recent years embedded computing has emerged as the new paradigm for the design and implementation of modern computer systems, succeeding mainframes, minicomputers, and finally desktop computers. They consist in the fastest growing market share for computing products, already accounting for the largest number of systems being deployed. As shown by Ebert and Jones [1], in 2009 for each person living in a developed country there were about 30 embedded processors, clearly exceeding the number of any other kind of computing system. Typical examples of embedded systems include tablets, smart phones, digital cameras, TV set-top boxes, security systems, personal robots, automotive control devices, and medical equipment. In addition, as systems requirements grow in complexity, the need for highly specialized hardware/software platforms becomes more critical.

A design option of growing interest is the use of a *heterogeneous computing platform*, which may contain possible combinations of CPUs, DSPs, and custom devices and accelerators based on reconfigurable hardware. A typical example of such architecture is the MORPHEUS platform [2]. As shown in that article, the platform was shown to be viable for the successful implementation of complex embedded systems such as wireless telecommunications, video processing, networking, and smart cameras. Those implementations also stress the need for a robust toolset, as the programming model for a heterogeneous platform can be quite complex. In the case of the MORPHEUS platform, the toolset is based on the MOLEN paradigm, which allows general purpose conventional code and hardware descriptions to coexist in a program. This programming model was originally employed by the MOLEN Polymorphic Processor, another example of heterogeneous architecture [3]. Heterogeneous computing

also stresses the need for multitarget programming frameworks, able to reduce the required implementation effort. The *Accelerator System* [4] is an example of this kind of programming system, being able to target multicore processors with vector instructions, GPUs, and FPGAs using a single source code described in conventional high level languages, such as C++. Liquid metal [5] is a similar tool, also targeting CPUs, GPUs, and FPGAs but using Lime as source code, a new programming language.

As it happens during any technology shift period, skills shortage can be a problem as the current curriculum may not address the whole set of issues involved. According to Ricks et al. [6], developers need to have a detailed knowledge about devices and applications in order to master the current embedded system sophistication. The range of skills required for embedded systems design encompasses knowledge about digital logic, computer architecture, hardware description languages, real-time requirements, and resource-aware software development. Although these topics are adequately taught in computer science courses, students tend to see them as isolated units, with little relation to embedded systems. Reconfigurable computing can be used as a common platform for teaching all of those subjects and also to expose students to some of the main concepts and practices in the field.

Reconfigurable hardware, such as FPGAs (field-programmable gate arrays), is a rich and flexible platform, where students can create any kind of digital system, ranging from a single circuit to a very complex and heterogeneous embedded system architecture [7]. This is a good platform to teach students hardware/software codesign, allowing them to simulate and debug software and hardware together, to create or configure softcore microprocessors, and to optimize the system in terms of performance, resources usage, and power consumption. However, since programming hardware requires a different way of thinking than developing software, it is desirable that computer science degrees add more emphasis on teaching how to program systems with both components.

A well-recognized way to motivate students to learn about design, implementation, and programming of embedded systems is via contemporary and challenging applications, such as those found in the embedded mobile robotics field [8]. Practical-based approaches have been used by other educators as a way to catch student attention [9–11]. Mobile robotics is an emerging topic, and although significant advances have been observed in recent years, there are still many challenges to be faced, mainly when autonomous and customized robots are considered [12]. According to Touretzky [13], computer science students should be better prepared for the robotic revolution, since real robots are computationally complex, comprising both hardware and software processing cores.

As a result of studying mobile robotics, students learn reconfigurable computing and embedded systems, otherwise rather unpopular among many computer science students. However, the growing complexity of embedded systems design and applications makes real projects a far goal to the typical one-semester course. One way to bridge this gap is to foster practical teaching by means of *stronger interaction between classroom and research labs* at the university level. In this context, the main contributions of this paper are as follows:

(i) showing a way to reinforce undergraduate courses on embedded systems by using research results and expertise produced by graduate students, which can be seen as a strategy to put a project-based learning (PBL) pedagogical approach into practice

(ii) presenting real teaching experiences (at ICMC-USP, Brazil) and reasons for using FPGA platforms and mobile robotics to teach embedded systems in computer science programs

(iii) pointing out the main difficulties faced by the students with this methodology, which may help to reproduce and improve the strategies to be presented.

2. A Project-Based Learning Approach

The dynamic and multidisciplinary nature of computer science (CS), both in its core subjects and the ever-growing application domains, makes CS education particularly challenging. The primary approach to CS education is still based on knowledge transmission, which can be considered essential in some instances. However, it is also perceived as limited for a number of reasons, such as not promoting higher order thinking or advanced reasoning skills [14]. An alternative approach to CS education is project-based learning [15].

According to [16], *project-based* courses can be considered as a method within the *problem-based* learning toolkit, being common in computing education. Problem-based learning [17] seeks to empower students such that they take responsibility for their own learning. It was originally envisioned for medical education, a field where it is well established and is considered successful [18].

The difference between problem- and project-based approaches may be fuzzy in its principles and strategies. It can be argued that a *problem* is more loosely defined, while a *project* is a well-defined problem, with clear objectives. According to [19], a project can be considered as an instance of PBL if it adheres to the following five criteria.

(1) *Centrality*. Projects are central, not peripheral, to the curriculum contents and main objectives.

(2) *Driving Question*. Projects are focused on questions or problems that "drive" students to encounter the central concepts and principles of a discipline.

(3) *Constructive Investigation*. Projects involve students in a constructive investigation (goal directed process), involving inquiry, knowledge building, and resolution.

(4) *Autonomy*. Projects are student-driven to some significant degree, not being totally teacher-led, scripted, or packaged.

(5) *Realism*. Projects are realistic (not school-like), giving a feeling of authenticity to students.

A comprehensive analysis and categorization of practical problem-based learning in computing education can be found in [14]. It has been concluded in that article that

the penetration of problem-based learning into the average computing curricula is still shallow, with faculty members working in isolation. On the other hand, it was concluded that lecturers and students' experiences are usually positive. A more specific investigation on project-based course is reported in [16], focusing on course design, composition of groups, and assessment.

The educational approach described in this paper fits well in the five-criterion scenario defined by [19]: *centrality*, *driving question*, *constructive investigation*, *autonomy*, and *realism*. As such, it can be considered as a realistic case study of a project-based course in the later stages of a computer science degree.

Embedded systems have a complex nature when compared to desktop or server software, the prevalent platform adopted by most courses in the typical CS curriculum. A diverse mix of software and hardware components, functional and nonfunctional requirements, and significant differences between target applications make it difficult to adopt a conventional approach to embedded systems teaching. Previous experiences have shown to the authors that applying the conventional pedagogical approach to embedded systems teaching can be very effective for the particular scenario considered by the course. However, it was perceived that a significant number of students would feel reluctant to undertake a new project in embedded systems, even after a successful completion of the standard course. One of the reasons for that, the authors believe, is that those students would not feel confident enough to propose by themselves a significantly different design for the new project. By adopting a PBL approach and employing a hands-on and realistic project in the first course on embedded system, the authors aim to empower students with the skills necessary to tackle other complex and diverse projects in this field. This is a desirable outcome, considering that embedded systems development is an area of ever-increasing interest and presents a shortages of well qualified professionals.

The use of realistic projects in the course described in this paper is central to the adopted pedagogical approach. Rather than using the project as an illustration of theoretical concepts, it is a trigger for knowledge discovering and assembling of new and revisited techniques.

The remainder of this paper is organized as follows. The importance of reconfigurable hardware is discussed in the next section. Then, Section 4 argues about the importance of mobile robotics for the success of the approach presented in this paper. Section 5 presents the actual teaching scenario in which embedded systems are taught, along with the learning outcomes expected. Section 6 details the mobile robotic projects used in an undergraduate course teaching embedded systems. Section 7 presents the teaching platforms employed (hardware and software). Section 8.2 presents and examines the feedback given by students, followed by conclusions in the last section.

3. Reconfigurable Hardware

Field-programmable gate arrays (FPGAs) are reconfigurable hardware devices composed of an array of logic blocks, an interconnection network, and a set of I/O cells. The programmability of FPGAs comes from the configurability of these three elements in such a way that any computational function can be implemented [20]. The main FPGA advantage over dedicated devices—application specific integrated circuit (ASIC), which is very costly to be designed—is the flexibility of adapting the computational architecture according to the user needs after the silicon fabrication process is completed. However, to provide such flexibility, FPGAs come at higher cost in area, delay, and power consumption, when compared to ASIC devices [21]. In spite of those drawbacks, FPGA devices are gaining more and more importance over the years, as they bridge the gap between ASICs and microprocessors [22]. For this reason, they have been extensively used in both academic and industrial fields for prototyping new hardware architectures, for accelerating a wide variety of applications, and also for developing and deploying commercial products. However, knowledge about digital systems and hardware description languages is needed to efficiently program reconfigurable hardware, which can be considered a barrier for an even higher acceptance of this technology.

The best practices when designing for FPGAs include the description of the architecture in a high level description language (HDL), such as VHDL, Verilog, Bluespec, SystemVerilog, SystemC, or Handel-C. The textual description is synthesizable and is used as input to logic and RTL (register-transfer level) synthesis tools that automatically generate the circuit netlist. The netlist is then mapped, placed, and routed to create the bitstreams that are responsible for programming the target FPGA device. This development flow does not require full mastering of those methods and techniques, especially the ones related to logic synthesis and place and routing. In some way the flow can be seen as a black box, similar to the one used by students when compiling a software program to a target microprocessor. As it happens with software programming, to optimize or use some specific resources of the target system, students may need some knowledge about the underlying target architecture. Of course, if fully optimized hardware is required, more knowledge about hardware design, the underlying FPGA architecture, and the tools used in the development flow may be necessary.

University programs maintained by companies such as the one from Xilinx [23] and Altera's companies [24] allow the use of state-of-the-art commercial tools to program FPGAs. Despite of presenting some restrictions, simulators such as ModelSim [25] are also freely available for teaching purposes. Students can also download copies of those tools to use at home to work on homework assignments, thus improving their experience. The available EDA tools and FPGA platforms provide nearly every resource needed to master embedded systems design. Project development may have to accomplish the partitioning of software code in order to shift some processing tasks to dedicated hardware components. The first implementation may be a software-centric one, after which students may realize that hardware/software partitioning can be the only alternative to enforce system constraints (e.g., energy consumption, performance, etc.). By doing so, students can acquire knowledge related to Amdahl's Law [26],

profiling, identification of the computational structures more suited to be implemented as dedicated hardware components, interfacing schemes, communication overheads, and so forth.

To conduct the necessary practical experiments there is however the need to expose students to complex examples, allowing them to have a perception of the problems they might face in the real world. So a question may arise: what kind of applications shall students work on in order to acquire all the expertise that is needed to develop embedded systems? Bearing in mind the answer to this question, mobile robotics tasks and applications have been selected, as described in the next sections.

4. Why Mobile Robotics?

Developing mobile robots encompasses several challenges and can be a suitable set-up to teach students many aspects of modern and future embedded systems. Personal robots, for instance, are usually mobile and need a large degree of autonomy for localization, navigation, and mapping tasks. Many of the requirements usually presented by this kind of system are prevalent in almost all other kinds of the embedded system applications. More specifically, mobile robotics has been chosen as the target application for teaching purposes based on the following reasons.

(i) It includes all the characteristics of embedded systems, ranging from simple to complex ones.

(ii) It requires students to deal with concepts such as video, audio, sensors, wireless networking, motor control, and navigation.

(iii) It usually demands miniaturization, frequently leading to a system-on-a-chip (SoC) solution.

(iv) It accommodates many interesting requirements such as low cost, low energy consumption, and high performance.

(v) It needs flexibility and programmability.

(vi) It typically needs hardware/software codesign, since for most robotic kernels using software implementations alone does not permit achieving the required performance.

(vii) It is a very attractive topic to students.

(viii) All of those reasons are desirable when conducting a PBL course.

Based on previous and ongoing experiences conducted by our research group on this field, students seem to be very motivated to study subjects and work with examples targeting a mobile robot platform. Since embedded systems and reconfigurable computing expertise is acquired during the implementation of robotic kernels, the strategy can be considered as an important element to attract computer science students to topics otherwise not considered by them. Other authors have also successfully used robotics to teach concepts in digital design and in other courses as well [27, 28]. The work presented in this paper is similar to the ones in a broader context but differs significantly by the nature of the applications worked on by the students: Bindal's work focuses on a robotic arm, while Kim's work employs the well-known Lego Mindstorms kits.

Table 1: SSC0711: course schedule.

Week	Topics	
	Theoretical	Practical
1	Introduction: embedded systems	—
2	Embedded systems design flow	Tools
3	State machines	Candidate projects
4	Concurrent process	Problem analysis
5	Interfaces and protocols	SW implementation
6	Control systems	SW implementation
7	HW/SW partitioning	Performance analysis
8	HW/SW integration	HW modelling
9	—	HW implementation
10	—	HW implementation
11	—	HW implementation
12	SoC technology	HW implementation/tests
13	State-of-art HW/SW codesign	HW/SW integration
14	—	HW/SW integration/tests
15	—	Demonstration

5. Teaching Embedded Systems to Undergraduate Students

At ICMC-USP, one of the leading computer science departments in Brazil, mobile robotics has been used as practical study cases to teach embedded system theory and practice in an undergraduate, one-semester course, offered as part of the Computer Science Bachelors curriculum. The course is internally referred to as *SSC0711—Embedded Systems HW/SW Co-Design* (more information about this course (in Portuguese) can be obtained at https://sistemas.usp.br/jupiterweb/obterDisciplina?sgldis=SSC0711 or directly from the first author), running during 15 weeks. In terms of supervised teaching, the course is equally split between theory and practice (2 hours/week each). However, students spend significantly more time developing their chosen projects out of classes, receiving help from tutors whenever necessary.

Table 1 shows a summary of the course contents. The theoretical part of this course starts giving a brief review about ASICs and general purpose processors in the embedded system scenario. Performance, power consumption, development, and update costs are also analysed (week 1). Then, a typical embedded system design flow is presented (week 2), followed by HW/SW specification and modeling. Computation models are presented, including finite state machine, data flow, and Petri nets (weeks 3 and 4). The next topics give an overview about interfaces, protocols, and control systems (weeks 5 and 6). After this point, teaching is focused on process and development tools, showing how to identify HW/SW partitioning (week 7), design HW/SW integration, and synthesize the whole computing system on a SoPC, using FPGA technology (week 8). Different IC technologies for implementing SoCs are discussed (week

12). Finally, state-of-the-art work is presented regarding embedded system development, encompassing new tools, models, and integrated circuit technology (week 13).

In practical terms, the initial 4 weeks are devoted to the set-up of EDA tools and definition and analysis of the projects to be undertaken by each group (limited to 3 students). During weeks 5 and 7, students work on a preexisting software-only solution. The emphasis is on functionality and performance analysis, trying to identify possible bottlenecks to be improved by using hardware implementations. This leads to a SW/HW partitioning, modelling, and implementation of a hardware solution for selected processing kernels (weeks 8–12). Typically, the hardware implementation is in the form of a custom instruction for the Nios II processor. Integration and tests of the whole HW/SW solution are carried out in weeks 13-14, followed by the system final demonstration for evaluation purposes.

The core objective of this course is to teach hardware, software, and system design in an *embedded system context*. Students are allowed to take this course only after a series of prerequisites is met, which give them a solid background on digital logic, computer architecture and organization, and HDL languages. Students typically enroll for this course in their 4th year of studies. The practical projects used in the course are based on fundamental tasks used in autonomous mobile robotics: localization, mapping, and navigation, along with image processing algorithms for data extraction from the operating environment and for human-robot interface. The adopted algorithms are implemented as embedded systems based on FPGAs, with students being asked to achieve a set of goals for each problem, as defined by the assigned instructor. The learning outcomes (L_i) expected from those students after successfully developing such projects can be summarized by the 5 items shown below:

- L1: ability to design and implement custom embedded systems, in the form of SoCs;
- L2: ability to perform hardware/software codesign for a realistic project;
- L3: fluency in using a HDL for the development of a medium-complexity problem;
- L4: experience in using embedded operating systems, possibly an RTOS (real-time operating systems);
- L5: ability to engage on graduate-level research on custom embedded systems based on reconfigurable computing.

5.1. Course Overview. As already said, the course contents are equally divided between theoretical and practical activities. The robotic-based applications come as a complement to the material found in theoretical-orientated embedded system books, such as Marwedel [29], Wolf [30], and Vahid and Givargis [31]. They give students the essential knowledge about embedded systems, describing the main hardware and software components along with some design techniques. Given the theoretical background needed to develop embedded systems, students start to work on practical experiments, customizing systems for robotic algorithms computations. The details of the assigned projects are presented in Section 6. The algorithms are initially provided to the undergraduate students as C/C++ code developed by graduate students during their research projects. As shown in Figure 1, the development flow for a practical project is divided into five steps.

(1) Run the C/C++ code on a PC platform, in order to better understand the algorithm implementation and also to gather code profiling information using *gcc* and *gprof* Linux tools. These first steps aim to identify the most demanding sections of code, by measuring the number of function calls and the time spent on each of them.

(2) Run the C/C++ code again on an embedded processor, in order to get new profiling information, comparing it with the previous run. That helps students to understand the main differences and limitations between a standard PC platform and an embedded processor (possibly a softcore such as Altera's Nios II).

(3) Reason and exploit the possibilities to meet the system requirements such as performance, resources usage, and flexibility. In this phase students basically engage on the hardware/software codesign, considering possible software optimizations, new custom instructions (implementing the corresponding custom hardware), multicore solutions, operating system tuning, and the development of new hardware components. In that case, they could operate as stand-alone blocks or as a coprocessor, either by directly being connected to a central processor or via a DMA channel.

(4) Analyse the performance and reliability of the new customized embedded system for both hardware and software modules, using software profiling tools such as *gprof* and hardware simulation tools such as ModelSim. Students have also the opportunity to validate their final projects in a real mobile robot, integrating it with on-going research developments undertaken by graduate students.

(5) Present the project to instructors and classmates, and write a technical report for evaluation purposes.

6. Practical Projects in Embedded Robotics

Every project used in the undergraduate course has already been deeply exploited by other graduate students from the department, as part of their MSc or Ph.D. degrees, as documented in these publications: [32–38]. This knowledge is then presented to the undergraduate students via papers or theses as *reference material*, which gives them a detailed description of the problem and corresponding algorithms, along with a reference design for comparison purposes. All of those materials, along with tutorials, tools, and source code are central to conduct the PBL course described in this paper. The next subsections describe the main robotic-based projects used so far in this course format. Each of those projects can be developed by a group of up to 3 students.

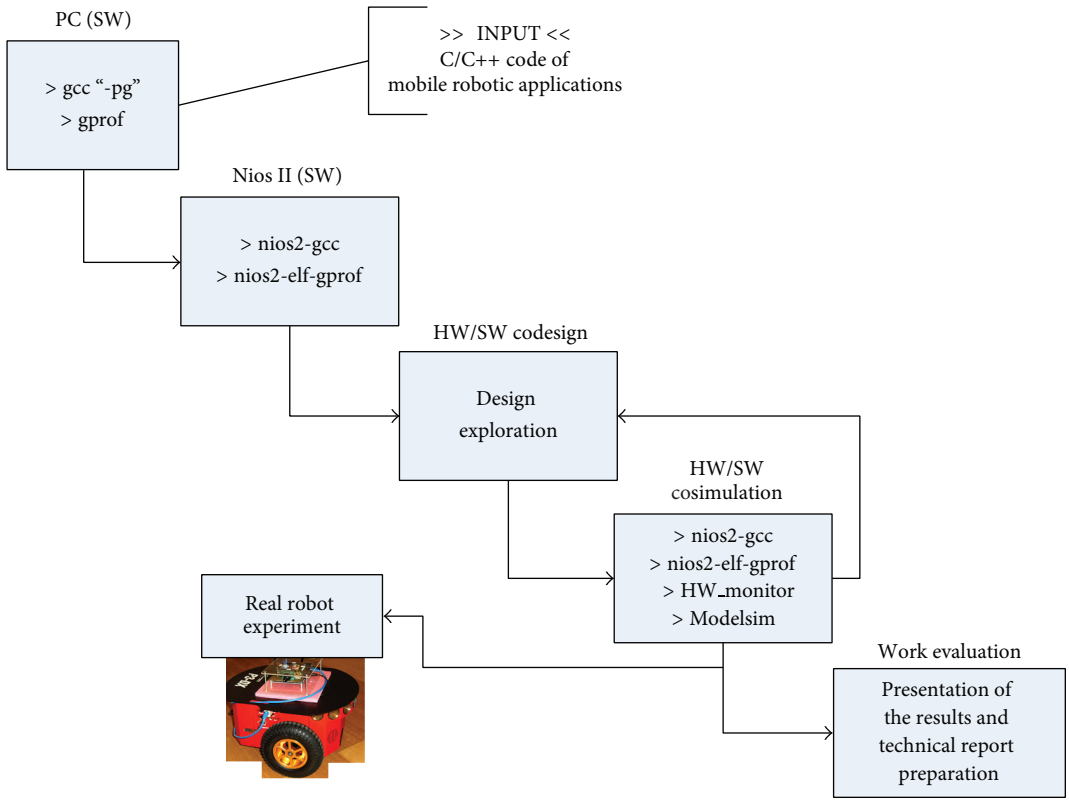

Figure 1: Embedded system development flow adopted by students.

6.1. Localization. The problem of autonomous robot *localization* can be solved using the *Monte Carlo localization* (MCL) algorithm [39], which uses a set of particles to estimate the real robot position inside a map constantly updated by sensor readings during its movements (most experiments are evaluated with offline data from test bench maps available from Radish [40]). This algorithm is quite interesting, since each particle can be evaluated independently of each other, allowing the students to exploit a high level of parallelism in hardware. As the number of particles scales, subject to a set of conditions such as sensor type and environment characteristics, balancing the computing hardware to meet system requirements is an interesting challenge. The easiest solution would be to code the whole MCL algorithm in C and run it in an embedded processor (software), while the hardest would be to code it in a synthesizable HDL and then generate the custom hardware. To find the balance between those two alternatives is a typical hardware/software codesign problem. In addition to design options influencing performance, the students also need to pay attention to some qualitative aspects of the system, such as random number generator supporting the MCL algorithm and the numerical precision regarding floating to fixed-point conversions. Bonato et al. [32] present the MCL algorithm implemented on an FPGA using the Mersenne Twister pseudorandom number generator, where a good balance between hardware and software is achieved. The design described in that paper is presented to the students and used as a guide for developing and evaluating their solutions.

6.2. Mapping. In mobile robotics a *mapping* algorithm aims to create a map of the navigation environment. This problem can be subdivided in two classes: when mapping is performed with knowledge of the robot position and when the position is estimated simultaneously during the mapping process. The latter case is known as *simultaneous localization and mapping* (SLAM) problem, with most solutions being based on probabilistic inferences derived from Bayesian filters involving high computational complexity and a large volume of data. The *extended Kalman filter* (EKF) algorithm [41] is a classical Bayesian derived solution, able to build a feature-based map without previously knowing the robot position. Bonato et al. [33] presented the first SoPC (system-on-a-programmable chip) of the EKF applied to the SLAM problem, which has been used as reference by students during the course. Differently than MCL, the implementation of this method exhibits high data dependency, basically consisting of multiplication and addition of matrices. In this case, parallelism exploitation is more complicated, as the students need to look more carefully to the problem in order to find how to split the internal operations without violating the code semantics. To better understand how to optimize the matrix operations on hardware, students are encouraged during the course to investigate state-of-the-art solutions, such as those presented by Irturk et al. [42].

6.3. Navigation. The *navigation* problem consists of defining a way for the robot to achieve its destination following some

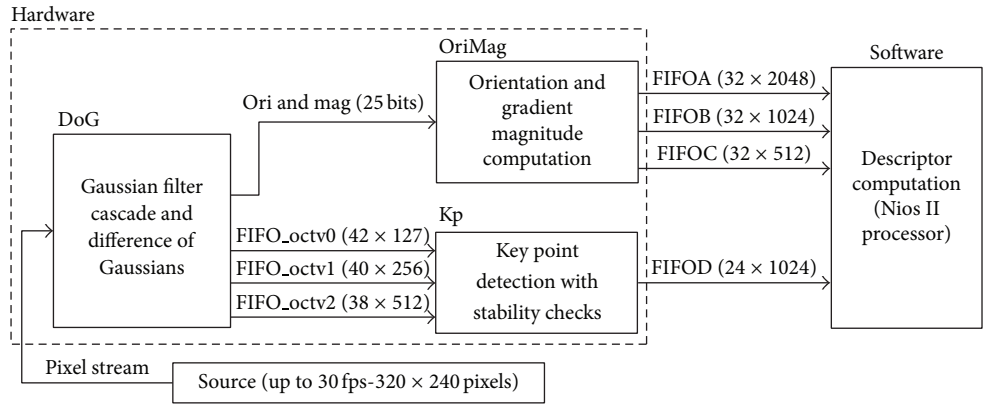

Figure 2: A hardware/software embedded system codesign comprising of three hardware blocks to extract features from images and precompute data for the descriptors and one software block to associate descriptors with features; the blocks communicate via dedicated channels along with FIFOs to store temporary data while receiver blocks are busy [34].

strategy, such as the fastest, the safest, or the easiest path. A quite simple algorithm used for navigation is *potential fields* (PF). In spite of presenting some limitations, such as trapping the robot in a particular place due to local minima [43], it is a well-known solution used in mobile robotics. In this method the navigation strategy is to reach a destination without collision, with obstacles being the *repulsive forces* and the destination being the *attraction force*. Due to its simplicity, a software-only implementation of the PF algorithm is not as interesting as a design option in an embedded system course. However, a hardware implementation poses some challenges, in particular a set of trigonometric functions that can be difficult to be implemented as custom hardware. For this reason, students are first asked to optimize the software by solving trigonometric functions based on Taylor series and then try to solve the problem in hardware using CORDIC algorithms and fixed-point representation. The hardware solution is similar to the one previously presented in [32], where a hardware implementation of the CORDIC algorithm is integrated into the system as custom instruction as well.

6.4. Image Processing. An interesting application in terms of students motivation and learning outcomes is the *scale-invariant feature transform* (SIFT) algorithm [44], which is considered one of the most robust solutions to extract features from images. This algorithm has been applied to many areas. In mobile robotics mapping it can be used as a data source to build feature-based maps from images. Its implementation as an embedded system is quite complex, specially if a high frame processing rate is required (e.g., 30 frames/second). A software-only implementation, such as running the application on the softcore Nios II processor at 150 MHz, would take more than one hour to extract the features from a single input image of 320 × 240 pixels. However, hardware and software customizations can lead to an embedded system able to process the input image in 1/30 seconds. For instance, a typical SIFT algorithm configuration needs to apply 18 different Gaussian filter kernels over an image in order to extract a feature candidate (known as key point). This throughput requirement represents 18.1 M pixel/s, which can be achieved by an efficient pipeline architecture, as demonstrated in Bonato et al. [34]. As the entire SIFT algorithm is too complex to be implemented by one student group during a semester, the problem is split between a number of groups, with each of them having to meet performance targets for its corresponding assignment. Figure 2 demonstrates the whole SIFT algorithm divided in blocks, with each block being assigned to a group of students.

6.5. Auxiliary Resources. In order to validate the embedded system produced, students are encouraged to integrate their system into a real *Pioneer 3DX robot* [45] running the *Player* control software [46]. Instructors provide students with a client for this software running on an embedded processor, in order to mitigate the integration task [35]. Additionally, students are given a set of auxiliary cores (in VHDL/Verilog) to support the project implementation, such as the IEEE-754 FPU (floating-point unit) module, an AMBA bus controller (ARM standard), an Ethernet module (MAC 10/100), hardware performance monitors, and a pseudorandom number generator based on the Mersenne Twister algorithm. Students are also allowed to get other cores from third-party providers, for instance, from the OpenCores website (http://opencores.org/). We believe this kind of support is particularly important to give a "real world" aspect to the course and would be infeasible without close interaction between undergraduate and graduate projects.

7. Teaching Platforms and Tools

FPGAs are a suitable alternative to introduce many concepts in labs and to let students implement real systems. That is mainly due to their programmability, availability of intellectual property (IP) cores for easy interfacing to external devices, availability of softcores that program part of the FPGA resources as a microprocessor, and availability of embedded hardcore components (DSPs—digital signal processors, memory blocks, and high speed I/O ports). All those features are extensively used throughout the course and lab projects presented in this paper. In addition, they are

widely available at a relatively low cost, with good supply of development tools. This is a desirable feature for an eventual replication of the PBL course described in this paper.

7.1. FPGA Boards. Typical FPGA boards include the necessary physical interfaces to connect them to other components in order to set up an almost real operating environment. In our laboratory experiments we have adopted Altera DE2-70 development board, provided by Terasic company, equipped with a $4.3''$ LCD touch panel with resolution of 800×480 points and a 5 M pixel CMOS camera. The DE2-70 board features an Altera Cyclone II 2C35 FPGA and a myriad of other interfaces. Experience shows that, when students work with an environment close to what they are used to see on their gadgets (e.g., touch screen display from smart phones), there is an extra motivation for embracing exercises and projects related to embedded systems, since they get curious to understand how the actually system works internally.

7.2. Microprocessor Cores. FPGA companies provide RISC softcores that can be used in FPGA devices. Examples of such softcores are Xilinx's MicroBlaze, Altera's Nios II, and Aeroflex Gaisler's LEON. They also permit exploiting of multiprocessor systems by instantiating more than one softcore in the same FPGA. Those companies provide the integrated design environments needed to develop complex systems using hardware/software components. With those environments students can start with software solutions and then add cores to improve performance, for instance. Those environments include C compilers, debuggers, and simulators to program target microprocessors.

By practising such approach students can also acquire the required skills in computing system organization since they are real system architects during the development of their projects. The impact of cache, floating-point units, multipliers, and dividers on system performance is easily realized by students through execution. They study the impact on performance of the adopted scheme to couple dedicated hardware components to a microprocessor. To measure performance, students understand the need for hardware monitors. All of those features have proven to be valuable in the teaching process.

Using softcore processors allows students to acquire many embedded systems concepts. Memory mapped devices, input/output ports, existence or not of instruction and data memory caches, existence or not of an operating system, and existence or not of floating-point units are all concepts acquired by students when developing those systems. As the aforementioned, the reuse policy when developing tasks is also fomented. Students are also required to use modules already developed by others and to deal with the needed interfaces.

7.3. Custom Instructions and Coupling Hardware Modules. By using the Nios II microprocessor, students also study the impact of adding custom instructions to the softcore, which are used to improve the performance of time critical sections of code. With this approach, they realize the need and advantages of doing hardware/software codesign at the very beginning of the project. Facing students with the design of systems including both hardware and software components has shown to be a key element to tackle embedded system concepts.

7.4. EDA Tools. The practical experiments developed along the course can be conducted with either Altera or Xilinx's tools. However, in this course we have adopted the first option, since the development kit available in the laboratory is the DE2-70 featuring an Altera's FPGA. The tools used are SoPC Builder and Quartus II for hardware development and *nios2-gcc* and *nios2-elf-gprof* for software compilation, debugging, and profiling for the Nios II processor [24]. Additionally, Altera's ecosystem also provides the ModelSim tool for hardware and software simulation and hardware monitors for online performance analysis.

8. Evaluation of the Pedagogical Approach

In order to evaluate the effectiveness of the pedagogical approach described in this paper, two kinds of analysis were conducted: one is based on the instructor point of view, and the other is based on the student's perceptions and feedback. Those analyses are presented in the next subsections.

8.1. Instructor's Evaluation. We believe the desired learning outcomes listed in Section 5 are achieved, noticed via the final project quality. Most projects are as complete as the reference one, encompassing software and hardware on a single chip, accessing external and onchip memory, using embedded operating system or a single hardware abstraction layer, and also using external interface for debugging and for sending and receiving data to the embedded system. For instance, a group of students developed a project evolving the potential field algorithm where optimized mathematical operations for hardware were applied in such a way that the achieved results in relation to performance and area were better than the reference project adopted. This medium to high complexity problem prepares the students to face real project development.

Around 15% of the students that completed this course are working directly with embedded systems development in Brazilian companies and 20% have engaged in graduate course in the same area. What students have said is that the practical approach of the course gives them more confidence to face a job interview in the area of embedded systems and also to perform the actual work for those who get the job. We also have had the opportunity to analyse the students during their graduate course and we could notice that they were able to produce research results very soon when compared to their colleagues with other backgrounds.

8.2. Students' Feedback. We have collected responses to an anonymous questionnaire given to students. All the students have answered mobile robotics as the most important factor to motivate them to realize the embedded systems projects using reconfigurable computing technology. One of the

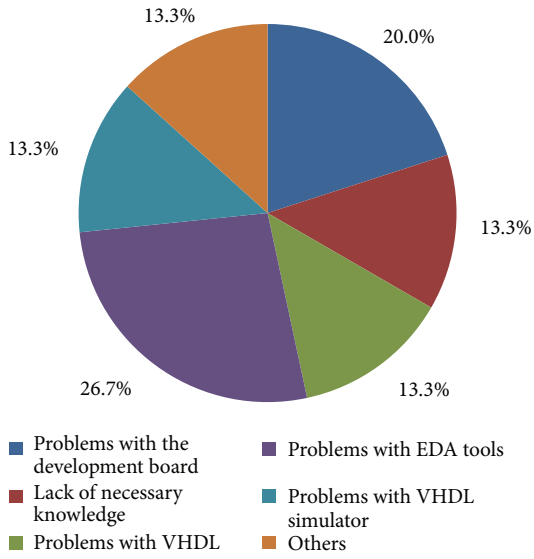

Figure 3: Percentage of student responses concerning the difficulties in concluding their projects.

reasons is that the students quickly realize the importance of the work to be done and see the results once it is implemented (e.g., image processing tasks). Based on the experiences we have had, their motivation is "application-driven," and they always pay more attention to and concentrate their energies on more visible and stand-alone projects.

The student responses concerning the aspects they had more difficulties when exposed to this teaching approach are shown in Figure 3. In general the main set of complaints is about the EDA tools and FPGA boards. The second set of aspects they refer more often are related to the VHDL simulator, the VHDL language, and the lack of some specific background knowledge. Some of the students indicate the memory limitations of the board as the main difficulty to implement some projects. Another important difficulty factor pointed out by students is the problems related to developing C programs when no operating system is used. In this context, they refer to the problems associated with allocating and reallocating memory as the most difficult one. Some of them also refer to the rather long initial time needed to begin a project, which is strongly related to the long learning curves needed to master the initial skills on using the tools and the VHDL language.

In terms of the skills students recognize they should have been acquired before, 50% of them mention advanced digital design as the most important one. About 10% of them refer to the need for learning more about computer architecture. These two aspects reinforce the importance of having courses in computer architecture where the design of a microprocessor is fully understood (including pipelining issues) and not only courses explaining assembly languages and superficial computer architecture concepts. About 30% refer to the need of learning more about applications (control systems, digital signal processing, image processing, etc.). Only one student referred to the importance of knowledge on parallel and multiprocessing systems.

Concerning the VHDL language, a very high percentage of students (about 50%) refer to its concurrency model as the most difficult issue to master. About 30% refer to the global semantic of the VHDL language as a difficult property to master. One of them refers to the previous and long experiences in software imperative programming as one of the reasons for the difficulties. Strangely, another one mentioned that his previous experience in designing hardware using schematics did not help and created some confusion in the first time using VHDL. According to the responses, students do not find the syntax and structure of the language a problem but reveal the new programming paradigm as the most difficult issue. One student complained about the lack of good tutorials, while another one about the complexity of the language and the not so clear and understandable subset of VHDL behaviours for synthesis. Those complaints were somehow expected, given the major areas those students are undertaking (either computer science or computer engineering).

The increasing importance of heterogeneous architectures has led to a number of projects aiming to develop compiling environments targeting multiple executing platforms, such as CPUs, GPUs, DSPs, and even FPGAs [47]. Those tools use in source code either well-established programming languages such as C/C++ [4] or special purpose languages such as Lime [5]. Hardware and FPGAs targets are of particular interest and have been the object of several efforts trying to automatically bridge the gap between high level programming languages and hardware custom structures [48]. Although several research tools have been developed and made available, we believe they are currently not appropriate to be used in a first course in embedded systems for undergraduate students, as they may abstract important aspects that should be exposed to them. Nonetheless, some of those tools and technologies are reaching mainstream use, such as Xilinx Vivado Design Suite [49], and should be adopted at least partially in our course, as those kinds of tools are gaining increasing importance when targeting heterogeneous platforms. Another promising alternative is Altera SDK for OpenCL [50], a framework allowing software programmers to write kernel functions in OpenCL C, to be compiled for execution on FPGA targets. The typical use of such system is mixing hardware accelerated functions with host code running on a CPU (e.g., ARM Cortex A9).

Concerning the level of difficulties of the projects, about 56% of the students reckon that the difficulty level is similar to the most difficult problems they have experienced during their academic trail. The remaining students consider the difficulty level of the proposed projects to be average, when compared to other assignments. Interestingly, none of the students considered the projects as being the easiest ones during his/her undergraduate or graduate program.

8.3. Ongoing Improvements. Current research work in the department is concerned with an EDA tool dubbed *RoboArch* being developed by MSc students. RoboArch is a component-based tool for developing hardware architectures for mobile robots [36]. The tool supports system modelling based on the MARTE profile [51] and simulation in electronic system level (ESL) integrated into real/simulated mobile robot platforms

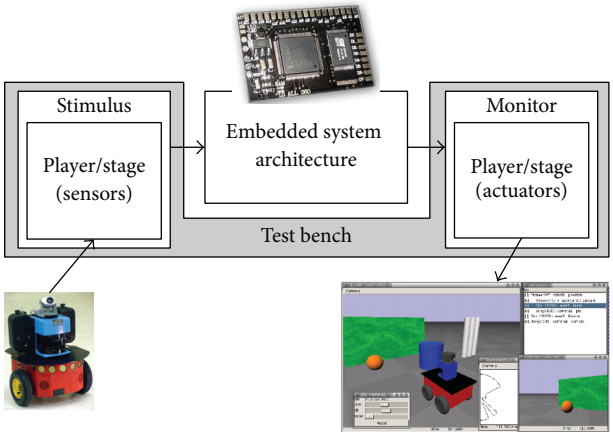

Figure 4: A hardware-in-the-loop simulation environment for embedded mobile robotics development.

(see Figure 4). When available, the tool should allow for easier access to more realistic experiences with robotics. The components will be stored in a repository using IP-XACT pattern [52], facilitating the exchange of hardware and software kernels among students. Considering that ESL simulation can be more than 3 orders of magnitude faster than register-transfer level (RTL) simulation, undergraduate students benefit from this by modelling and debugging whole embedded system in a feasible time, even before having the actual hardware implemented. Having the ESL model validated, the hardware part of the system can either manually or automatically be converted into RTL (Verilog/VHDL), depending on the ESL language adopted. When available to undergraduate students, we believe it should improve the development flow as a whole and so the students motivation and commitment to the assigned projects. As already pointed out, the use of HLS tools is an option which has been considered to complement developments based on VHDL language. In particular, some experiments are being carried out to evaluate the suitability of Xilinx Vivado [49] for the course purposes.

9. Concluding Remarks

This paper has shown experiences in using mobile robotic kernels to teach embedded systems based on reconfigurable hardware. Specifically, FPGAs and related EDA tools are being used so that students can realize systems with hardware and software components. Mobile robotic kernels accommodate most embedded system characteristics and requirements, and students seem to have higher levels of motivation to accomplish the kind of tasks that have been proposed in this area. A crucial aspect of the methodology is the close interaction between teaching and related research activities, which has allowed the development and testing of rather complex projects, not otherwise feasible using standard textbooks. This strategy effectively implements a project-based learning approach, which is specially desirable towards the end of a student's graduation period. After some years of developing and applying this methodology, some satisfactory results have been achieved. That is revealed by the increased interest shown by students to follow research projects related to embedded systems, the improved performance of graduates working in this area, and also some feedback from former students working in the industry. It should be noticed that students do not need deep skills in robotics to implement robotic kernels for their projects. However, some of the students feel so motivated that they learn by themselves many concepts related to mobile robotics, which is obviously very positive and has encouraged the authors to refine and continue using the PBL pedagogical approach described in this paper.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to acknowledge the partial support given by the bilateral cooperation CNPq/Grices (Brazil/Portugal) program that permitted missions from both groups to Brazil and Portugal during 4 years and thus the exchange of experiences and knowledge between the authors. The authors also would like to thank FAPESP for the financial support given to publish this work.

References

[1] C. Ebert and C. Jones, "Embedded software: facts, figures, and future," *Computer*, vol. 42, no. 4, pp. 42–52, 2009.

[2] N. S. Voros, M. Hubner, J. Becker et al., "MOrpheus: a heterogeneous dynamically reconfigurable platform for designing highly complex embedded systems," *Transactions on Embedded Computing Systems*, vol. 12, no. 3, article 70, 2013.

[3] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. L. M. Bertels, G. Kuzmanov, and E. M. Panainte, "The MOLEN polymorphic processor," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1363–1375, 2004.

[4] S. Singh, "Computing without processors," *Communications of the ACM*, vol. 54, no. 8, pp. 46–54, 2011.

[5] J. Auerbach, D. F. Bacon, I. Burcea et al., "A compiler and runtime for heterogeneous computing," in *Proceedings of the 49th Annual Design Automation Conference (DAC '12)*, pp. 271–276, ACM, New York, NY, USA, 2012.

[6] K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "An embedded systems curriculum based on the IEEE/ACM model curriculum," *IEEE Transactions on Education*, vol. 51, no. 2, pp. 262–270, 2008.

[7] S. Vassiliadis and D. Soundris, *Fine-and Coarse-Grain Reconfigurable Computing*, Springer, Berlin, Germany, 2007.

[8] T. Bräunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*, Springer, Berlin, Germany, 2nd edition, 2006.

[9] C. S. Lee, J. H. Su, K. E. Lin, J. H. Chang, and G. H. Lin, "A project-based laboratory for learning embedded system design with industry support," *IEEE Transactions on Education*, vol. 53, no. 2, pp. 173–181, 2010.

[10] B. Benson, A. Arfaee, C. Kim, R. Kastner, and R. K. Gupta, "Integrating embedded computing systems into high school and early undergraduate education," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 197–202, 2011.

[11] P. Bertels, M. D'Haene, T. Degryse, and D. Stroobandt, "Teaching skills and concepts for embedded systems design," SIGBEDRevelation 6, 4:1–4:8, 2009.

[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, Cambridge, Mass, USA, 2005.

[13] D. S. Touretzky, "Preparing computer science students for the robotics revolution," *Communications of the ACM*, vol. 53, pp. 27–29, 2010.

[14] M. J. O'Grady, "Practical problem-based learning in computing education," *ACM Transactions on Computing Education*, vol. 12, no. 3, article 10, 2012.

[15] J. S. Krajcik and P. C. Blumenfeld, *The Cambridge Handbook of the Learning Sciences*, Cambridge University Press, Cambridge, UK, 2006.

[16] D. Richards, "Designing project-based courses with a focus on group formation and assessment," *ACM Transactions on Computing Education*, vol. 9, no. 1, article 2, 2009.

[17] D. Boud and G. Feletti, *The Challenge of Problem-Based Learning*, Kogan Page, London, UK, 1997.

[18] J. Strobel and A. van Barneveld, "Whem is pbl more effeective? A meta-synthesis of meta-analyses comparing pbl to conventional classrooms," *Interdisciplinary Journal of Problem-Based Learning*, vol. 3, no. 1, pp. 44–58, 2009.

[19] J. W. Thomas, "A review of research on project-based learning," 2000, http://www.bie.org.

[20] C. Bobda, *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications*, Springer, Berlin, Germany, 1st edition, 2007.

[21] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2007.

[22] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospectiv," in *Proceedings of the conference on Design, Automation and Test in Europe Table of Contents*, pp. 642–649.

[23] Xilinx Inc., ISE Design Suite, 2014, http://www.xilinx.com.

[24] Altera Corporation, Quartus II Web Edition, 2011, http://www.altera.com.

[25] Mentor Graphics, ModelSim—Advanced Simulation and Debugging, 2011, http://www.mentor.com.

[26] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[27] A. Bindal, S. Mann, B. N. Ahmed, and L. A. Raimundo, "An undergraduate system-on-chip (SoC) course for computer engineering students," *IEEE Transactions on Education*, vol. 48, no. 2, pp. 279–289, 2005.

[28] S. H. Kim and J. W. Jeon, "Introduction for freshmen to embedded systems using LEGO mindstorms," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 99–108, 2009.

[29] P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, Springer, Berlin, Germany, 2nd edition, 2011.

[30] W. Wolf, *Computers as Components, Second Edition: Principles of Embedded Computing System Design*, Morgan Kaufmann, San Francisco, Calif, USA, 2008.

[31] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, John Wiley & Sons, New York, NY, USA, 1st edition, 2002.

[32] V. Bonato, B. F. Mazzotti, M. M. Fernandes, and E. Marques, "A mersenne twister hardware implementation for the Monte Carlo localization algorithm," *Journal of Signal Processing Systems*, vol. 70, no. 1, pp. 75–85, 2013.

[33] V. Bonato, E. Marques, and G. A. Constantinides, "A floating-point extended kalman filter implementation for autonomous mobile robots," *Journal of Signal Processing Systems*, vol. 56, no. 1, pp. 41–50, 2009.

[34] V. Bonato, E. Marques, and G. A. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 12, pp. 1703–1712, 2008.

[35] D. E. Wolf, J. A. Holanda, V. Bonato, R. Peron, and E. Marques, "An FPGA-based mobile robot controller," in *Proceedings of the 3rd Southern Conference on Programmable Logic (SPL '07)*, pp. 119–123, February 2007.

[36] V. Bonato and E. Marques, "Roboarch: a component-based tool proposal for developing hardware architecture for mobile robots," in *Proceedings of the 4th IEEE International Symposium on Industrial Embedded Systems (SIES '09)*, pp. 249–252, July 2009.

[37] M. C. Sacchetin, J. J. Lopes, D. F. Wolf, J. L. Silva, and E. Marques, "Analysis and implementation of localization and mapping algorithms for mobile robots based on reconfigurable computing," *Latin American Applied Research*, vol. 37, no. 1, pp. 31–34, 2007.

[38] J. Silva, M. M. Fernandes, V. Bonato, R. Menotti, J. M. Cardoso, and E. Marques, "Using mobile robotics to teach reconfigurable computing," in *Proceedings of the IEEE Computer Society Workshop on Reconfigurable Computing Education (WRCE '06)*, p. 6, Karlsruhe, Germany, 2006.

[39] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*, vol. 2, pp. 1322–1328, Detroit, Mich, USA, May 1999.

[40] A. Howard and N. Roy, "Radish: the robotics data set repository," 2003, http://radish.sourceforge.net.

[41] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167–193, 1990.

[42] A. Irturk, B. Benson, S. Mirzaei, and R. Kastner, "GUSTO: An automatic generation and optimization tool for matrix inversion architectures," *Transactions on Embedded Computing Systems*, vol. 9, no. 4, article 32, 2010.

[43] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '91)*, pp. 1398–1404, Sacramento, Calif, USA, April 1991.

[44] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[45] Adept Technology Inc, Research Robots, 2011, http://www.mobilerobots.com/ResearchRobots.aspx.

[46] B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhatme, and M. J. Matarić, "Most valuable player: a robot device server for distributed control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '01)*, pp. 1226–1231, Maui, Hawaii, Maui, November 2001.

[47] D. F. Bacon and S. Singh, "Compiling complete programs into circuits (ccpc 2012)," in *A Workshop Co-Located with ASPLOS*, London, UK, 2012.

[48] J. M. P. Cardoso and P. C. Diniz, *Compilation Techniques for Reconfigurable Architectures*, Springer, Berlin, Germany, 1st edition, 2009.

[49] Xilinx Inc, Vivado Design Suite, 2014, http://www.xilinx.com.

[50] Altera Corporation, Altera SDK for OpenCL, 2014, http://www.altera.com.

[51] OMG, UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, 2009, http://www.omgmarte.org/.

[52] Spirit Consortium, IP-XACT v1.4: a specification for XML meta-data and tool interfaces, 2008, http://www.spiritconsortium.org/home.