Educational Case Studies with an Open Source Embedded Real-Time Java Processor

Rasmus Ulslev Pedersen
Embedded Software Laboratory
Dept. of Informatics
Copenhagen Business School
Howitzvej 60
2000 Frederiksberg, Denmark
rup.inf@cbs.dk

Martin Schoeberl
Institute of Computer Engineering
Vienna University of Technology
Austria
mschoebe@mail.tuwien.ac.at

ABSTRACT
In this paper we show a platform which allows for education and training of a number of essential embedded skills. The Java optimized processor (JOP) is open source and has been used in several educational and training sessions and we cover how each setting has trained a special skill set. The experience covers basics from undergraduate education to Ph.D. level education. At each level different properties of the system are emphasized. Our emphasis on the interdisciplinary of embedded systems education is based on referenced research findings. This way we provide empirical findings and couple it with academic frameworks.

1. INTRODUCTION
Training professionals and students on embedded systems programming and design touches two fields: educational learning theory and embedded programming.

According to [10], the design of embedded systems is multi-disciplinary and inter-disciplinary. The skill set, we can add, will also require the students, as well as professionals, to collaborate if they are to develop an application with a complex human computer interface or something involving a streaming data source, for example. With pair programming, and especially using Java, we have a programming language with APIs that span from embedded platforms such as JOP [19] to desktop and server machines. The Java platform spans from the Java Micro Edition (J2ME) to the Java Enterprise Edition (J2EE).

The setup demonstrated in this paper is suited for introductional programming in Java as is offered in many textbooks such as the one by Morik and Klingspor [11]. Often we have a combination of human computer interaction and software engineering terms to address in these classes. Clemmesen and Nørbjerg combine the terms we use in this endeavor [2]. Another study concludes that students arrive at a study program in their freshman year with different prerequisites [7]. The setup in this paper, where an embedded system like JOP is programmed with Java and VHDL, is both very easy and very complex at the same time. Some embedded educators prefer reconfigurable hardware [17]. There are other interesting virtual machines suitable for education like *leJOS*.[1] In terms of the study by Kautz and Kofoed, we can point to the possibility of using the embedded Java virtual machine also provided by the *leJOS* Sourceforge project or JOP[2] to make it more interesting for students with different backgrounds to start in a computer program or for corporate training of professionals. Lego Mindstorms NXT is an open source platform for embedded programming and it is also possible to teach the small operating system TinyOS on it [13]. In this way, the students or professionals with previous (embedded) system programming experience can program directly on the embedded JVM and those without previous experience can just take the PC solution. However, a FPGA-based Java processor is the focus in this paper.

The paper outline is as follows: we start with a short technical overview (from an educational point of view) of the Java Optimized Processor (JOP) in Section 2. We cover key properties of JOP and relate them to training of students and professionals. Then we cover several actual case studies of using JOP for teaching at a non-technical university in Section 3. This demonstrates the approachability of JOP even for people with a different background. We list how JOP is used at a technical university in Section 4. Finally, we discuss the experiences and provide the conclusion for the paper.

2. THE JAVA OPTIMIZED PROCESSOR
The Java processor JOP [19] is a hardware implementation of the Java virtual machine (JVM). JOP is open source under the GNU GPL. Thus, it is freely available for educational purposes. All tools needed to build the processor, the application, and the library are freely available. Therefore, this system is an ideal teaching vehicle. JOP can be used as a platform for low-level hardware exercises, system level exercises within the implementation of the JVM, and real-time application development for embedded Java. An interface to Lego Mindstorms enables to build Java controlled Lego robots. The most popular platform for JOP is the Cyclone

[1]`http://lejos.sourceforge.net/`
[2]`http://www.jopdesign.com/`

WESE'09, October 15th, 2009, Grenoble, France
Copyright 2009 ACM 978-1-XXXXX-XXX-X/09/10 ......$10.00.

Figure 1: The FPGA used for hosting JOP.

FPGA from Altera as shown in Figure 1. It is available from `www.jopdesign.com`.

2.1 JOP: Under The Hood

Java bytecodes are translated into microcode instructions or sequences of microcode. The difference between the JVM and JOP is best described as the following: The JVM is a CISC stack architecture, whereas JOP is a RISC stack architecture. JOP is implemented in a field programmable gate array (FPGA). One design goal for JOP is its applicability to worst-case execution time (WCET) analysis. This design principle is consistent throughout the JOP processor. As the processor is implemented in an FPGA and all sources are available, it is also possible to add specialized hardware units [23]. From a training point of view this is interesting as the JOP processor covers everything from easy-to-understand Java to hardware specification in VHDL.

2.2 Thread Scheduling on JOP

The thread scheduler on JOP is preemptive with fixed priorities. On a control switch a complete stack belonging to the active thread is saved. Each frame on the stack contains the saved program counter (PC) that points to the next instruction that will get executed once control returns from the invoked method. JOP can be used to educate and train professionals and students in multi-threaded programming. The threading model of JOP is intended for the future standard on safety-critical Java [4]. Therefore, the students are exposed to a system for real-time programming.

2.3 JOP's Processor Pipeline

JOP is a fully pipelined architecture with single cycle execution of microcode instructions and a novel approach to mapping Java bytecode to these instructions. Three stages form the JOP core pipeline, executing microcode instructions. An additional stage in the front of the core pipeline fetches Java bytecodes – the instructions of the JVM – and translates these bytecodes into addresses in microcode. Bytecode branches are also decoded and executed in this stage. The second pipeline stage fetches JOP instructions from the internal microcode memory and executes microcode branches. Besides the usual decode function, the third pipeline stage also generates addresses for the stack RAM. As every stack machine instruction has either pop or push characteristics, it is possible to generate fill or spill addresses for the following instruction at this stage. The last pipeline stage performs ALU operations, load, store and stack spill or fill. At the execution stage, operations are performed with the two topmost elements of the stack.

Since JOP has a rather simple pipeline it thus becomes even more interesting from an educational point of view. In other processors the pipeline stages can be many more, leading to unnecessary clutter in a learning scenario. Furthermore, the simple pipeline allows for tight estimation of the worst-case execution time (WCET) of Java programs [21]. In one class the students have to program a Lego robot with at least two threads, perform WCET analysis with the provided tool, and perform the resulting schedulability analysis.

2.4 JOP in Industry

Several real-world projects in industry are based on JOP [18]. The first project with JOP was a distributed control application in the railway domain. On a distance of up to 1 km the contact wire is tilted up to simplify loading and unloading of goods wagons. Each mast is equipped with one JOP powered embedded system that controls the asynchronous motor. The program was written in a simple cyclic executive style to simplify the safety argument. Further projects followed: a remote terminal unit for supervisory control, an industrial lift controller, and a support system for single track railway control. Most of the applications are under open-source and part of the JOP distribution. Therefore, students can see how real-world applications are written in embedded Java.

2.5 Available Material

The complete source of JOP and the documentation is online available. A good start is the main web site of JOP:

`http://www.jopdesign.com/`

and a Wiki platform that is editable by the JOP users at:

`http://www.jopwiki.com/`

A reference handbook is available as PDF and as printed book [20], which comprehensively covers the JOP processor in terms of tool chain and PC setup. It is a point where most newcomers to JOP will have to start. Furthermore, teaching slides are available from the main web site[3] with sources when applicable. This makes JOP an interesting choice for many embedded courses in industry or in higher education.

Martin Schoeberl, the creator of JOP, leads a growing community surrounding JOP. He manages online forums in the form of a Yahoo mailing list.[4] This community consists mainly of two groups: people playing with JOP just for fun and students using JOP as basis for their Master's thesis or PhD thesis (e.g., [3]).

3. AT A NON-TECHNICAL UNIVERSITY

In this section, we describe how JOP has been used at the Dept. of Informatics, Copenhagen Business School. The department bought 20 JOP boards to have enough for all stu-

[3]`http://www.jopdesign.com/teaching.jsp`
[4]`http://tech.groups.yahoo.com/group/java-processor/`

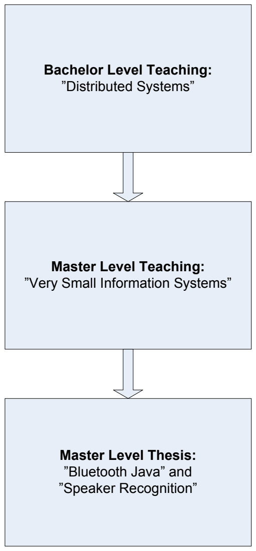


Figure 2: Using JOP at a non-technical University.

dents. We found the open community of JOP was helpful for the students to find information.

It is an inter-disciplinary study program that we used the JOP board for. The study program features about equal amounts of computer science, organization, and economics. Embedded systems are found everywhere and it is therefore important that we train students and professionals outside the traditional profiles in this kind of systems as well.

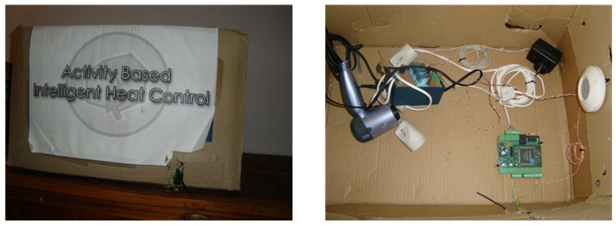
The JOP board has been used for bachelor level teaching, master level teaching, and master level theses at this non-technical university. Figure 4 shows the relation of the different courses at CBS. We will discuss the experience with JOP for educational purposes below.

3.1 Bachelor Level Class

The JOP board was used for a class called *Distributed Systems*. There is teaching material available online from this class. This class covers topics such as networking with the IP protocols. In this class the board's Ethernet controller was used to let the embedded board serve as a small webserver. It can serve a simple HTML page, which gave the students a basic understanding of how this would work for an embedded system. The relevance comes from seeing embedded systems as small Internet connected devices. Furthermore, the board was used for exercises that generated UDP packets, and these packets were collected with a network sniffer such as the open source freely available sniffer called Wireshark.

3.2 Master Level Classes

The JOP board has been used for two classes at the master level. One was called *Distributed Data Mining*. The other class was on embedded systems and information systems. The goal was to create small systems which served a purpose in terms of making parts of a house more intelligent for example. This class was called *Very Small Information Systems*. The VSIS class is a 15 ECTS class requiring 450 student work hours. For the exam, the students created a



(a) Box from Outside (b) Box from Inside

Figure 3: Example student setup for the activity based heating project.

so-called product as described above.

There were a number of student projects and the format of the class was also to let the students write a real article in academic format about the project they did with the JOP board. Each article was reviewed by a peer group and by the teachers as well. We found this approach very productive, and the final articles were better than we expected at a business school. We also experimented with Wikiversity[5] as an interactive teaching platform during this course.

The projects were quite different. They ranged from a barcode recognizer to an intelligent heat controller (see Figure 3).

Intelligent Humidity and Temperature Controller This group did a project to address the problem of remote monitoring of humidity levels in wooden walls. It addresses a problem of monitoring a wooden house while it is not occupied as is the case for holiday residences or summer houses. The group used a humidity controller to collect data and then they used a data mining program called WEKA to make the decisions using an XML feed over the internet.

Activity-based Intelligent Heat Regulation This group set up the JOP FPGA board with infrared movent detection sensors. The registered activity is then analyzed and the predicted activity is used to pro-actively control the heat in the house. An overall aim of this project was to save energy at it would be cheaper to heat the house only in the rooms where it was known that activity would take place at certain times of the day. Figure 3 shows the experimental setup built by the students.

Barcode Recognizer In this project, the JOP board was used to recognize a bar code. The group made a system where a picture was taken with a mobile phone, then transferred to the Java board. It then used algorithms such as Fisher discriminant analysis to recognize the EAN-13 barcodes.

Automatic Speaker Verification The sound is collected to a wave file format, which is then recognized by the JOP board. Here the UDP protocol is used to transfer the wave file to the JOP board. The group applied (on

[5]`http://en.wikiversity.org/wiki/School:Very_Small_Information_Systems`

the JOP board) a Hidden Markov Model and a nearest neighbor algorithm to the data.

Stock Data Analysis This group analyzed several algorithms on a PC with a data mining package to see which one provided the best tradeoff between accuracy and speed. They found a simple algorithm that fitted the limited constraints of an embedded system, and implemented it on JOP.

This class was a success as we trained a semi-technical group of students at a business school to use an embedded system. The group had no knowledge about embedded systems before the class, and it was possible to get them started with this in just a few weeks. The students got exposed to the differences between embedded systems programming and programming on a desktop system – especially the resource constraints and the small available library on JOP.

We used a setup of two people. One with deep knowledge of embedded systems (Martin Schoeberl), and one with interdisciplinary knowledge of data mining and embedded systems (Rasmus Ulslev Pedersen [12]).

3.3 Master Level Theses

The JOP board has been used for a master thesis project on Bluetooth. Some of the students took the master level class to get started with the JOP board (see Section(3.2)). The board was equipped with a Bluetooth radio connected to JOP via the serial line. The student who completed the thesis implemented code on the JOP board to be able to exchange data via the OBEX protocol between the embedded JOP based system and a mobile phone.

A second thesis was conducted with the aim of identifying words recorded in real-time from a microphone. The group needed to create a custom PCB [9] for recording the words and to this end they received assistance from an outside company. This custom PCB was plugged into the expansion slots of the JOP board itself.

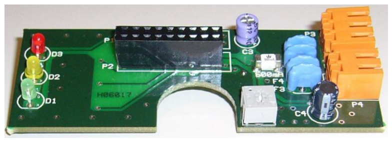


Figure 4: A Master Thesis Speech Recognition PCB.

4. AT A TECHNICAL UNIVERSITY

At the Vienna University of Technology JOP is used in several settings by Martin Schoeberl. From a special course on the JVM, for Bachelor and Master projects, up to research in the context of Ph.D. theses.

In Vienna, we have built a group of students around that project and have regular meetings. An interesting point of that group of students, even at different levels, is that the interaction among the students last longer than a single semester.

4.1 The JVM in Hardware

At the Vienna University of Technology a specialized course, the *Java Virtual Machine in Hardware*, has been given over the last years. The students learn the inner working of the JVM and tradeoffs between hardware and software implementation of system functions. The course is a combination of given talks and lab work. The practical part of the course consists of group work on a selected hardware or software problem for embedded Java. The introduction course on the JVM and JOP let several students to continue on embedded Java with their Bachelor or Master's thesis.

The course covers the whole embedded stack from processor design, system programming (the JVM), library implementation, up to the application design. The students can decide on which level they will perform the project. The more successful projects are included in the main distribution of JOP. The fact that the laboratory work contributes to an open-source project is highly motivating for the students. In the following list, a few project examples from the course are given:

Long Bytecodes for JOP Long operations (64-bit) are expensive in hardware. Two students implemented the JVM long operations on top of the available 32-bit instructions. The implementation was done in Java and included an extensible test suite for the operations.

The JVM Instruction `idiv` in VHDL for JOP Integer division was enhanced by two students moving the implementation from software to hardware. They implemented a division module in VHDL.

CACAO on a MIPS A group of Ph.D. students, taking this master level course, solved a more challenging task: to compare the performance of JOP against a JIT based JVM they started to port the open-source JVM CACAO [8] to the open-source, soft-core MIPS processor YARI.[6] As a follow up to this project we extended that work on using a JIT compiler for real-time systems and published a scientific paper [1].

Java Locks with PCE The current implementation of `synchronized` on JOP is just turn off the scheduling interrupt. That is effective priority ceiling emulation (PCE) without setting the ceiling level (default is maximum). One student adapted the real-time scheduler on JOP to acknowledge the ceilings of objects on synchronized blocks. This project involved the student in practical application of real-time scheduling theory in a Java context.

Scoped Memory Scoped memory is a simplified version for dynamic memory management for real-time systems to avoid garbage collection. One student implemented scoped memory, as defined by Safety Critical Java. The implementation involved adaption of JVM system code.

Wireless Robot Control The Lego robot, powered by JOP, was extended with an XBee module for remote control via a simple protocol. This project involved using a

[6] `http://repo.or.cz/w/yari.git`

solder iron and adapting the VHDL code of JOP at a high level (configuring a second serial port and assigning the correct pins).

Table/Lookupswitch in Microcode Bytecode instructions `tableswitch` and `lookupswitch` are implemented in Java on JOP to save microcode memory. To simplify WCET analysis and to speedup those operations two students implemented the two bytecodes in JOP microcode. The project covered the very internal layer of the microcoded implementation part of the JVM.

WCET Analysis Using the UPPAAL Model Checker This project produced a first prototype of a model checking based WCET analysis for Java bytecodes. One student continued with this topic towards his Master's thesis [5].

JOP User Interface One group implemented a user interface for JOP. This project included VHDL design of the mouse and keyboard interface, a VGA controller, and the user interface Java library. Furthermore, the JOP simulator was adapted for easier development of the graphics library.

NFS with ejip Implementation of the network file system (NFS) client on top of the ejip UDP/IP stack. Emphasis on this project was to provide a small, time-predictable implementation. This project is now continued as a Bachelor project.

The examples show that individual groups selected different layers of the full hardware/software stack for their project. Some groups even decide to cover several levels of the embedded stack.

4.2 Bachelor Projects

Several Bachelor projects are based on JOP. In the following two examples from quite different fields are given:

A Lego Robot Controller PCB Two students built a PCB to interface JOP with the sensors and actuators of the Lego Mindsstorms. The resulting PCB is shown in Figure 5. They built the electronics, interface hardware in the FPGA in VHDL, and a software library in Java for various sensors and actuators. Therefore, they covered the whole design stack of an embedded system. The interface is now used in the JVM in Hardware course to write small Lego robot programs, which have to be analyzed for its WCET.

Bytecode Optimizer Java bytecode, as omitted by the Sun `javac` compiler, is completely unoptimized. The optimization is left to the JIT compiler. However, for a Java processor optimization at bytecode level is an interesting option. This Bachelor work implemented the program analysis and optimization tool `Joptimizer`. The main speedup was gained by inlining short methods. `Joptimizer` uses a heuristic that contains the timing information of JOP to decide wether a method shall be inlined or not.

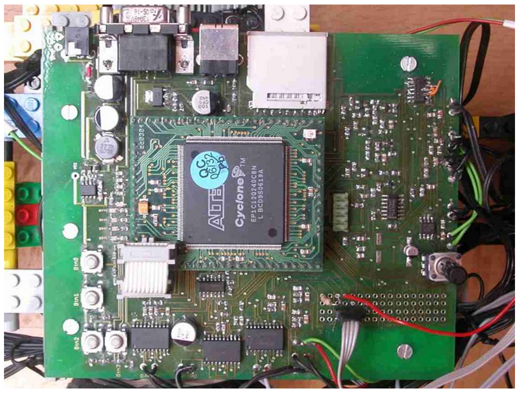


Figure 5: A PCB to interface Lego Mindstorms sensors and actuators.

4.3 Master Level Theses

As JOP is also used as a platform for research on real-time Java and real-time computer architecture, the students get involved with on-going research during their Master's thesis.

- Wolfgang Puffitsch ported picoJava, a competing Java processor from Sun, to an FPGA and compared it with JOP during his Master's thesis [15]. This thesis prepared well for the following Ph.D. study in the context of Java processors.
- Benedikt Huber continued after the *JVM in Hardware* course with a Master's thesis on WCET analysis. He redesigned the available WCET tool [21] and evaluated model checking for WCET analysis [5].
- Peter Hilber is currently implementing hardware transactional memory on a JOP based chip-multiprocessor system. The target of the work is an implementation that shall be analyzable for real-time systems.

4.4 Ph.D. Research

JOP is also used for Ph.D. studies. Christof Pitter finished his PhD thesis on a time-predictable JOP chip-multiprocessor (CMP) system [14] this year. The resulting CMP version of JOP is also used in the EU project JEOPARD [22]. Wolfgang Puffitsch uses the CMP version of JOP as basis for his PhD research on real-time garbage collection for multiprocessor systems [16]. Hi intends to finish his Ph.D. thesis next year.

5. DISCUSSION

We have found that JOP is well suited for education at higher level universities. JOP is more than a small embedded system. The fact that it also features an Ethernet controller makes it possible to use it in teaching that involves network communication as well. The real-time properties of JOP are appealing to students regardless of the needs for a real-time embedded system. They find it interesting to work with an embedded system where they can see that their Java code will be executed in a precisely known time.

The embedded Java system with JOP is a computer with very limited resources, e.g., the main memory is just 1 MB.

This limit of the main memory also results in a very restricted subset of the Java library (JDK). Therefore, students have to adapt their programming style for this constraint system. We are confident that this restriction provides a good preparation for the industrial needs of an embedded programmer.

JOP is based on the Java programming language. This language is very popular for teaching and it is one of the reasons that it was possible to have students experiment with the JOP board without too much introduction.

6. CONCLUSION

JOP as a flexible open source teaching platform for embedded systems has been presented in this paper. The processor is introduced and we argue why this FPGA-based Java processor is ideal for teaching and training of students and professionals. For example, the simple three stage pipeline is similar to other RISC processors, and the learning curve is approachable, but additional benefits apply to JOP because of the open source approach and the active community surrounding it.

In this paper we have showed several situations where the Java Optimized Processor have been used both in challenging embedded educational settings and for Master's and Ph.D. projects. Furthermore, we have shown case studies of using it for applications which are focused on intelligent embedded systems. Our observations and experiences are similar to those expressed at the First Workshop on Embedded Systems Education (WESE2005) [6]. For students and potential researchers it is an advantage that JOP is rooted in the academic environment.

Future work include experiencing with JOP in a distributed team-programming setting using the Eclipse platform. We plan to test a plugin for supporting JOP training and couple it with the open source XPairtise[7] Eclipse plugin. It will be an interesting team-programming platform for embedded systems education.

Acknowledgement

We would like to thank all our students for their sometimes enthusiastic approaches to their projects, their questions, their feedback, and the resulting discussions. Motivated students are the best motivation for teachers.

7. REFERENCES

[1] F. Brandner, T. Thorn, and M. Schoeberl. Embedded JIT compilation with CACAO on YARI. In *Proceedings of the 12th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009)*, Tokyo, Japan, March 2009. IEEE Computer Society.

[2] T. Clemmensen and J. Nørbjerg. Separation in theory, coordination in practice - teaching hci and se. *Software Process: Improvement and Practice*, 8(2):99–110, 2003.

[3] T. Harmon. *Interactive Worst-case Execution Time Analysis of Hard Real-time Systems*. PhD thesis, University of California, Irvine, 2009.

[4] T. Henties, J. J. Hunt, D. Locke, K. Nilsen, M. Schoeberl, and J. Vitek. Java for safety-critical applications. In *2nd International Workshop on the Certification of Safety-Critical Software Controlled Systems (SafeCert 2009)*, Mar. 2009.

[5] B. Huber. Worst-case execution time analysis for real-time Java. Master's thesis, Vienna University of Technology, Austria, 2009.

[6] D. J. Jackson and P. Caspi. Embedded systems education: future directions, initiatives, and cooperation. *SIGBED Rev.*, 2(4):1–4, 2005.

[7] K. Kautz and U. Kofoed. Studying computer science in a multidisciplinary degree programme: Freshman students orientation, knowledge, and background. *Journal of Information Technology Education*, 3:227–224, 2004.

[8] A. Krall and R. Grafl. CACAO – A 64 bit JavaVM just-in-time compiler. In G. C. Fox and W. Li, editors, *PPoPP'97 Workshop on Java for Science and Engineering Computation*, Las Vegas, June 1997. ACM.

[9] M. Lundsgaard and J. K. Rasmussen. Jopspeech: Embedded java speech recognition sdk. Master's thesis, Copenhagen Business School, Dept. of Informatics, 2006.

[10] P. Marwedel. Embedded systems education: how to teach the required skills? Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, 2004. Session Moderator-Marwedel, Peter and Session Panelist-Gajski, Daniel and Session Panelist-De Kock, Erwin and Session Panelist-De Man, Hugo and Session Panelist-Sami, Mariagiovanna and Session Panelist-Söderquist, Ingemar.

[11] K. Morik and V. Klingspor. *Informatik kompakt*. Springer, 2005.

[12] R. U. Pedersen. *Using Support Vector Machines for Distributed Machine Learning*. PhD thesis, Dept. of Computer Science, University of Copenhagen, 2005.

[13] R. U. Pedersen. Tinyos education with lego mindstorms nxt. In J. Gama and M. M. Gaber, editors, *Learning from Data Streams. Processing Techniques in Sensor Networks*, chapter 14, pages 231–241. Springer Berlin Heidelberg, September 2007.

[14] C. Pitter. *Time-Predictable Java Chip-Multiprocessor*. PhD thesis, Vienna University of Technology, Austria, 2009.

[15] W. Puffitsch. picoJava-II in an FPGA. Master's thesis, Vienna University of Technology, 2007.

[16] W. Puffitsch. Decoupled root scanning in multi-processor systems. In *CASES '08: Proceedings of the 2008 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 91–98, New York, NY, USA, Oct. 2008. ACM.

[17] F. Salewski, D. Wilking, and S. Kowalewski. Diverse hardware platforms in embedded systems lab courses: a way to teach the differences. *SIGBED Rev.*, 2(4):70–74, 2005.

[18] M. Schoeberl. Application experiences with a real-time Java processor. In *Proceedings of the 17th IFAC World Congress*, Seoul, Korea, July 2008.

[19] M. Schoeberl. A Java processor architecture for

[7] `http://xpairtise.sourceforge.net/`

embedded real-time systems. *Journal of Systems Architecture*, 54/1–2:265–286, 2008.

[20] M. Schoeberl. *JOP Reference Handbook: Building Embedded Systems with a Java Processor*. Number ISBN 978-1438239699. CreateSpace, August 2009. Available at `http://www.jopdesign.com/doc/handbook.pdf`.

[21] M. Schoeberl and R. Pedersen. WCET analysis for a Java processor. In *Proceedings of the 4th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2006)*, pages 202–211, New York, NY, USA, 2006. ACM Press.

[22] F. Siebert. JEOPARD: Java environment for parallel real-time development. In *Proceedings of the 6th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2008)*, pages 87–93, New York, NY, USA, 2008. ACM.

[23] J. Whitham, N. Audsley, and M. Schoeberl. Using hardware methods to improve time-predictable performance in real-time Java systems. In *Proceedings of the 7th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2009)*, Madrid, Spain, September 2009. ACM Press.