Hardware Platform Design Decisions in Embedded Systems - A Systematic Teaching Approach

Falk Salewski
Embedded Software Laboratory
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
salewski@informatik.rwth-aachen.de

Stefan Kowalewski
Embedded Software Laboratory
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
kowalewski@informatik.rwth-aachen.de

ABSTRACT

Designers of embedded systems can choose between a large variety of different hardware platforms. The question often arising is which hardware platform is suited best for a certain application. This decision is usually made by an expert in industry being familiar with a variety of hardware platforms. Therefore, it is of major interest how this expert knowledge and the skills necessary for such a selection process can be taught to students. In this paper a systematic hardware platform selection process based on hardware attributes is presented. Moreover, it is proposed how to integrate this approach in the embedded systems education.

Categories and Subject Descriptors

K.3.2 [**COMPUTERS AND EDUCATION**]: Computer and Information Science Education

Keywords

Embedded System Education, Hardware Platform Selection, CPU vs. PLD, MCU vs. FPGA, Microcontroller, Reconfigurable Hardware

1. INTRODUCTION

Embedded systems integrate hardware and software components and require developers with skills in both subjects. Hardware skills should include the capability of adopting a systematic approach to making design decisions in choosing between various hardware platforms for a given embedded systems application. These decisions are nontrivial because there is a large number of quite different hardware platforms available. These are CPU based systems such as microcontrollers (MCU) and digital signal processors (DSP), as well as Programmable Logic Devices (PLD)[1] as Complex Programmable Logic Devices (CPLD) and Field Programmable Gate Arrays (FPGA). For some applications also combinations of different discrete devices or an integration of a microprocessor core in an FPGA is preferable. Moreover, the available hardware devices are constantly changing over time. Thus, embedded systems education needs to include, in addition to knowledge of basic functional properties of current hardware platforms, the capability to systematically analyze both functional and non-functional features of hardware devices and the skills to map these to the requirements of a given specified application to be designed.

One way to impart some of these experiences to students is by conducting exercises and lab courses dealing with different hardware platforms and architectures, as for example done in [2] or [8]. This hands on learning is very useful to teach students the fundamental properties of different platforms, mostly with respect to functionality. However, we experienced in our own embedded lab course, presented in [8], that not all influencing factors can be taught this way. Additionally, a comparing overview of different hardware platforms is needed. Promising approaches can be found for example in [9] or [10]. Both give a good comparative introduction into CPU and PLD based embedded systems. Furthermore, an example illustrating the trade off among different hardware architectures is given in [10]. However, a systematic approach for hardware platform selection is missing. In this paper we are going to propose such an systematic approach for hardware platform selection and discuss how it could be used successfully in embedded system education.

The remainder of this paper is organized as follows. In section 2 the selection process is analyzed in order to gain the information necessary for further work. The steps of a proposed hardware platform selection process are presented in section 3. An integration of this approach in the embedded systems education is proposed in section 4, followed by a discussion in section 5. Finally, a conclusion is given in section 6.

Note: All physical parts of an embedded system are referred to as hardware (e.g. MCU, FPGA), while the languages which determine the behavior of these systems are referred to as software (e.g. assembly, C, VHDL).

2. AN ANALYSIS OF THE SELECTION PROCESS

In industry, the selection of hardware platforms for individual applications is a common task. The decision is made on the basis of experiences from former projects and the

[1]PLDs are also known as *reconfigurable hardware*

WESE '06, October 26th, Seoul, South Korea
.

knowledge of domain experts and is influenced by many factors. First of all, the hardware platform must allow to realize the desired *functionality* given by functional requirements. Hardware attributes that define functional requirements are the *functional range* and, in case of real-time systems also the *performance*. Functional range defines the amount of functionality (e.g. number of functions, size of data structures) that could be realized on a certain hardware platform. The attribute performance defines how fast the functions are executed.

Besides functional requirements non-functional requirements are of interest for the selection process. They do not represent the required functionality itself but additional qualities the later system must have. Typical non-functional requirements range from reliability, maintainability and testability to power consumption and marketability.

A lot of publications are dealing with non-functional requirements in software design, e.g. [6] or [7]. The strong interaction between software and hardware components in embedded systems demands that designers deal with the combination of both components. The selection of hardware platforms is usually prior or parallel to software design. In order to build an overall system fulfilling the requirements, it is not only important to consider hardware properties, but also their influence on certain software properties.

Accordingly, the selection process demands knowledge of hardware properties, including properties influencing the software, and skills to map these properties on the requirements of the system to be designed. In the following, we will present an approach that can support this selection process.

3. SYSTEMATIC HARDWARE PLATFORM SELECTION PROCESS

3.1 Connecting System Qualities with Hardware Properties

As a result of the former section, functional and non-functional requirements are important for the hardware platform selection process. As a structure representing how much a system fulfills non-functional requirements, a tree structure as the *quality attribute utility tree* used in the ATAM technique [6] developed at the SEI for software qualities, seems appropriate. Since we are interested in support for hardware platform selection, a structure representing the influencing hardware properties on the corresponding system qualities (utility to fulfill non-functional requirements) and functionalities (utility to fulfill functional requirements) is needed. According to these considerations the quality attribute utility tree has been modified to a *hardware attribute tree* (see Fig. 1) representing hardware attributes which influence functional and non-functional properties of the corresponding system.

At this stage it is necessary to find out how these hardware attributes are influenced by hardware properties. This question is quite obvious for some properties while other require the knowledge of experts.

Another aspect is the way these dependencies are presented. We decided to provide them in a graphical way. This presentation, which can be found for several attributes in the figures 2 to 9, is done on an abstract level which allows us to remain mostly unspecific with respect to concrete hardware platforms.

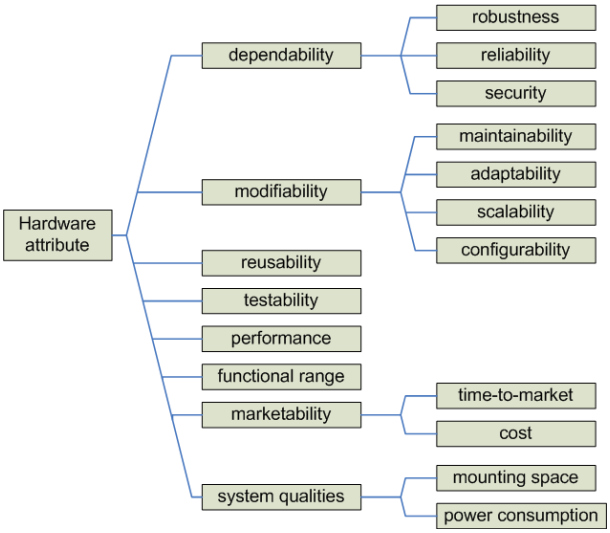

Figure 1: hardware attribute tree

As an example, the attributes *performance*, *functional range* and *marketability* and their influencing hardware properties are looked at closer in the following.

The attribute *performance* is influenced by the architecture of the processing unit (what can be done in one cycle?) and the clock frequency (how fast is one cycle?), as depicted in Fig. 2. The architecture of the processing unit itself is influenced by the bit width (how many bits can be changed at a time?), by the available instructions (what can be processed within one instruction?), by the grade of parallelism (how many things could be done in parallel?), and the integrated peripherals (Which operations have dedicated hardware modules?). Especially the last aspect is of major interest in embedded systems.

As can be seen in Fig. 3, the *functional range* of a device is determined by the available memory (e.g. how much program code could be stored) and/or the available chip area (e.g. how much functionality can be synthesized?), depending on the type of hardware platform. The functional range is also determined by the integrated peripherals (e.g. allow dedicated hardware modules parallel execution of certain functionalities?), the I/O capabilities (can all required external devices be connected?) and abstraction capabilities (how is complexity handled?). The latter factor might be considered controversial, since lack of abstraction capabilities does not limit functional range in principle. However, possibilities to use hardware abstraction or hierarchies are useful to improve the efficiency of many design processes [9].

In contrast to the former two attributes, *marketability* is influenced by factors which are mostly outside the actual target hardware. One of the influencing factors is *time-to-market*. The development time of a device should be kept short in order to save money and to achieve advantages over competing companies. Hardware properties influencing *time-to-market* are the general development effort of a functionality on a certain type of hardware platform (how long does it take to implement this functionality on this type of hardware?), the expertise of the development team (does the team have experience in this type or a comparable type of hardware?) and the external design support. The

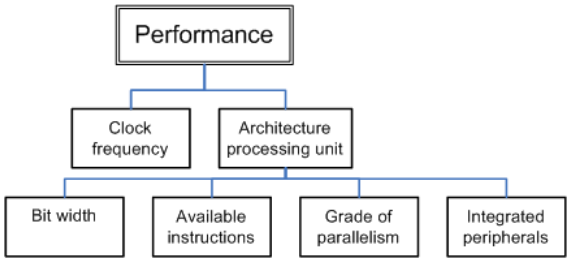

Figure 2: Hardware attribute performance

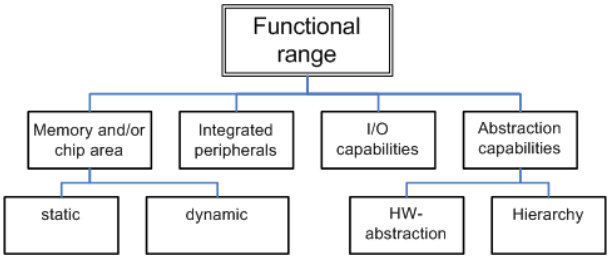

Figure 3: Hardware attribute functional range

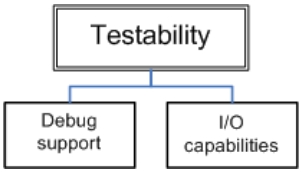

Figure 4: Hardware attribute testability

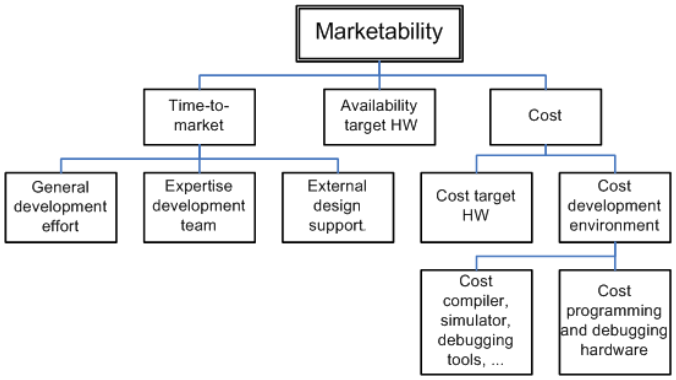

Figure 5: Hardware attribute marketability

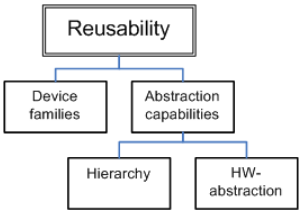

Figure 6: Hardware attribute reusability

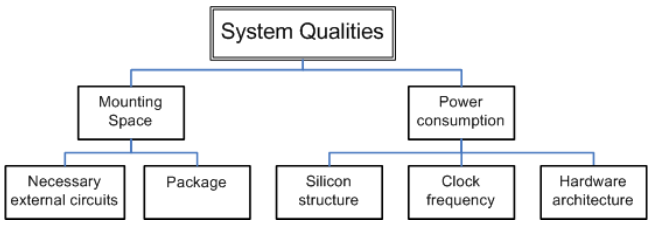

Figure 7: Hardware attribute system qualities

support can be further divided in hardware and software design support. Both include factors as the quality of available tools and the quality of available hotlines, newsgroups, and implementation examples. Another important factor influencing the marketability are the *costs* resulting from a certain hardware platform decision. First of all, these costs include the according target hardware platform itself and the costs to integrate this device physically in the system (e.g. special layout requirements). The degree of influence to the selection process depends mainly on the number of units which should be produced as well as on the target market. The costs for a hardware platform in a high volume consumer product (e.g. mobile phone) are probably of higher influence than the corresponding costs in an unique industrial plant. Another factor are the costs of the development environment needed for a successful implementation of the desired functionalities. This environment includes software as compilers, simulators and certain debugging tools, and hardware as programming and debugging devices. Finally, the availability of a certain hardware platform is an important factor for the marketability. If a design is developed for a certain hardware and the production of this device is stopped, a lot of design effort will be lost if the system cannot be migrated to another hardware platform easily.

The hardware influences presented in the structures above can be divided in *direct* and *indirect* influences. The hardware attribute *robustness*, for example, is influenced directly by hardware properties as the silicon structure or the I/O capabilities. This is different for the hardware attribute *time-to-market*. This attribute is influenced by factors as for example the expertise available in the design team. This expertise includes knowledge about the hardware (direct hardware influence), but also abilities of developing the software for this device (indirect hardware influence). These indirect influences integrate software design issues in the hardware design process, as for example postulated in [3].

Due to the abstraction, applied for hardware platform independence, this representation lacks of concrete information regarding individual hardware platforms. Since this information is important for the selection process, additional information might be provided in form of tables which will be presented in the following section.

3.2 Detailed Hardware Properties

For a successful selection of the optimal hardware platform, it is important to have detailed information about the individual hardware properties (as e.g. available instructions). This information could be taught, for example, on

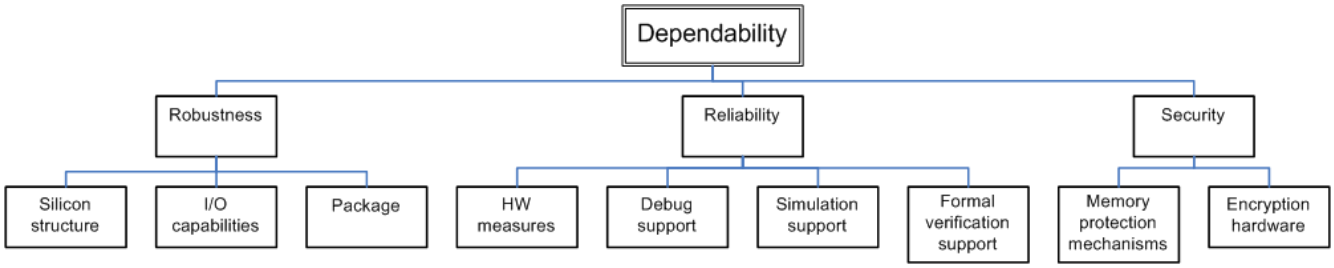

Figure 8: Hardware attribute dependability

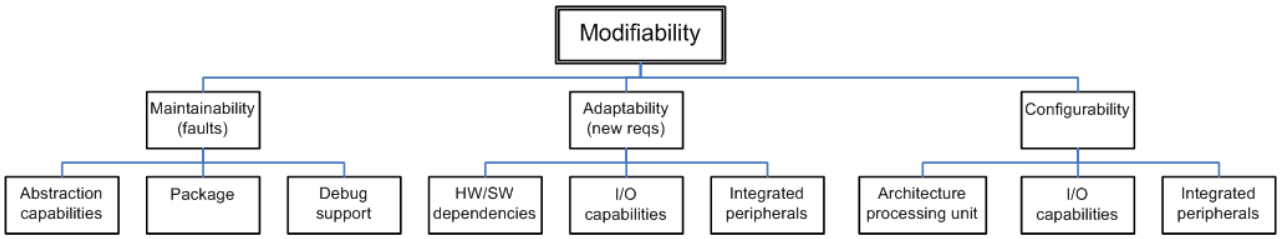

Figure 9: Hardware attribute modifiability

basis of subject-specific literature as [9] or [10]. In our opinion, a useful way to provide a survey of these information is in form of tables as this structure allows to emphasize similarities and differences of different types of hardware platforms. As an example, table 1 represents information regarding the hardware properties influencing the attribute *adaptability*. In the table structure, information of each hardware property is divided into three subcategories. A first category contains general information applicable for all hardware platforms. A second and third category each contain information specific for one of the two main groups of embedded systems, CPU based systems and programmable logic devices respectively. This presentation allows to get information about similarities and differences of these two groups very quickly. Further subcategories can be added, if necessary, by starting a description with the according device name (e.g. DSP: special algorithms for data streaming). Only general information should be included in theses tables in order to maintain clarity. For further information it should be referred to relevant publications (e.g. paper, book chapter, data sheet).

An alternative to the table structure could be a HTML based structure which would also allow the integration of additional levels of hierarchy. To be able to print a readable paper on basis of this structure (e.g. for a lecture script) an according print option has to be integrated to arrange the different levels of hierarchy in a paper readable format.

3.3 Hardware Platform Selection

Since functional and non-functional requirements are important for the hardware platform decision process, a thorough requirements engineering process should be performed in advance (see e.g. [1]). After this process the important qualities and functionalities for the system under consideration are known. In case of programmable systems these requirements have to be met by the combination of hard- and software. For hardware, the according hardware attributes could be found in the graphical structure presented in section 3.1. For each attribute, influencing hardware properties could be found with help of this structure. More detailed information of these properties with respect to different hardware platforms could be found in a table structure as presented in section 3.2 which should be supplemented by information from specific data sheet.

On basis of this information a structured hardware platform selection is possible. Trade-offs usually will be necessary since the different hardware attributes often compete with one another. It is important to mention that the structure presented should not provide the optimal solution but should form a basis to systematically compare the available options.

3.4 Example: 4 channel frequency generator

In order to illustrate the proposed approach, we present the following example. The desired system is a programmable 4 channel frequency generator that is to be programmed via an RS232 serial connection (12 byte message + CRC), with a minimum time between messages of 100ms.

The description of the task includes several functional requirements as performance and functional range. These have to be used for the selection process. In order to have complete requirements, an analysis of the non-functional requirements has been done whose results can be found in table 2.

In this example we analyze the following hardware platforms (with respect to requirements rated as medium or high):

1. low cost MCU
2. medium size MCU
3. low cost MCU + small CPLD
4. medium size FPGA

Table 1: Hardware properties influencing adaptability

Factors influencing adaptability	CPU	PLD	Describtion
HW/SW dependencies	X	X	Determines the effort necessary to transfer software from one device to another
	X	X	Transfer is necessary if requirements cannot be fulfilled with the actual device
	X		Microcontroller families ease migration from one MCU to another of the same family (microcontroller family = same CPU, different peripherals/packages/memory)
	X		Special application notes give help for migration process
	X		Hardware abstraction/operation systems could decrease dependencies between hardware and higher software layers
		X	If the functionality of a module is described in a hardware description language (**behavioral** description), the module can be transferred easily to any PLD suitable for this function In this case, transfer involves only a new assignment of package pins
		X	If the functionality of a module is described in **structural** description, the module can be transferred easily only to hardware platforms with a similar structure (e.g. the same basic elements used must be available)
I/O capabilities	X	X	Determines how easy new requirements with respect to I/O pins could be realized
	X		Usually, certain I/O functionality (e.g. serial input)) is mapped to a particular I/O pin
	X		Some I/O functionalities can be mapped to different I/O pins (e.g. analogue input)
	X		Few MCUs offer a free mapping of functionalities to I/O pins
	X		External buses (system bus, SPI, C2I,...) ease the integration of additional external peripherals/memory in the system.
		X	Usually, all I/O pins have the same properties/options (in, out, pull-up,...)
		X	Clock signals should be fed into the device via dedicated I/O pins
		X	Mapping of functionalities to I/O pins is done by software which allows maximum flexibility
		X	Pin assignment could influence chip area used for the design
Integrated peripherals	X	X	Multi purpose integrated peripherals increase adaptability
	X		Functionality of integrated peripherals can be determined via dedicated registers
	X		Integrated peripherals with a high number of options increase adaptability (and complexity)
		X	Almost all (digital) functionality is determined via software
		X	Predesigned modules are available for common functions, written in HDLs (*soft cores*, [4])
		X	Hard wired peripherals (e.g. clock divider) can be used if available
		X	Integrated peripherals usually do not include analog-to-digital converter
Additional influencing hardware properties can be derived from the hardware attributes *performance* and *functional range*			

Table 2: Example: non-functional requirements

requirement	priority
robustness	high
reliability	high
security	low
maintainability	low
adaptability	medium
scalability	medium
configurability	low
reusability	low
testability	high
time-to-market	high
cost	high
mounting space	medium
power consumption	low

3.4.1 Implementation 1: low cost MCU

First of all, it has to be checked if the functional requirements can be fulfilled with this hardware platform. We assume for this example that a suitable RS232 controller is integrated in this device. Accordingly, serial communication could be realized with minimum implementation effort and CPU burden (Read 12 byte from receive buffer, maximum every 100ms). Only the checking of the CRC would need some calculation time. The generation of the frequency signals is more challenging since no suitable on-chip peripheral is available. If it was the only job for the CPU, it could have been realized in software. However, the RS232 receive buffer has to be read and the CRC has to be checked each time a new message is arriving. The clock frequency of the MCU would have to be very high in order to generate four different signals doing this job concurrently. Programming would have to be realized on a very hardware specific level (probably assembly). Since the maximum clock frequency possible for this device is 16MHz, the task could not be realized on this device.

3.4.2 Implementation 2: medium size MCU

A faster MCU might be a solution for the problem stated above, or an MCU with an integrated peripheral suitable for the 4 channel frequency generation. In this second implementation we are analyzing the latter case. The MCU has to handle the incoming RS232 messages, the according CRC and the update of the peripheral for the frequency generation. Since most of the real-time tasks are done by on-chip peripherals, the MCU is able to handle the required functionality. As can be seen from Fig. 8, the system's *robustness* depends on the MCU hardware (check according data sheet) while the *reliability* depends on several factors. *Hardware measures* to improve the reliability could be a simple watchdog or more advanced built-in CPU- or memory-monitors. The *debug support* stands for the built-in debug hardware (JTAG, Trace, etc.) and the available software tools. They allow us to look inside the embedded system for verification purposes and are available for this device. *Simulation* and *formal verification support* permit the validation and verification of various functionalities before the software is executed on the target hardware platform. Simulation support is available for this MCU, but no real-time simulation. Since most of the real-time functionality is realized with on-chip peripherals this is not any drawback. Formal verification support is not available for this MCU. The *testability* is mostly determined by the debug support, which has been described above.

According to Fig. 9, *adaptability* is influenced by some hardware issues directly (How flexible are my internal peripherals) and by some indirectly (HW/SW dependencies, as how easy is it to migrate to another hardware platform if necessary?). Concerning the latter aspect, adaptability is probably improved if the hardware platform allows the use of higher programming languages. Further information can be found in table 1. In this case, *scalability* is similar to *adaptability* with an emphasis on direct hardware issues. Additional functionality is mostly limited by factors influencing *performance* and *functional range* (see Fig.. 2 and 3).

The influences of the hardware platform on the *time-to-market* are of indirect nature (Fig.. 5). The general software development effort is probably low, since the program has to initialize and coordinate on-chip peripherals mostly. The hardware development effort depends of the number and complexity of the external circuits needed to operate the hardware platform. In both cases, an influencing factor is the usability of the according data sheets and external design support as mentioned in section 3.1. As in all designs, the expertise of the design team has to be considered. The *costs* are determined by the cost for the individual target hardware platform and the costs for the according development environment. The planned production volume determines the importance of these two costs.

Finally, the *mounting space* is determined by the package of the target hardware platform and the amount of external circuits needed for operation (Fig.. 7).

3.4.3 Implementation 3: low cost MCU + small CPLD

In this third implementation, required functionality is realized on a combination of an MCU and a CPLD. The MCU handles the RS232 communication and the CRC check only while the performance-critical part of the frequency generation is done by the CPLD. The MCU and the CPLD are connected via 8bit address/data bus and 3 command lines. The frequency generator can be realized in the CPLD on basis of clock dividers. Functional requirements can be fulfilled by this implementation. The *robustness* of this implementation is depended on the MCU hardware, the CPLD hardware and their interconnection. The CPLD used in this example has damageable input circuits, according to limited protective measures (I/O capabilities). Accordingly, protective circuits for the four frequency signal inputs are necessary. The communication between the devices is an additional point of failure. The *reliability* of the application is now dependent on several hardware devices which decreases reliability in comparison to a single chip solution. On the other hand, the different functionalities (communication/frequency generation) are strictly separated from each other, which could reduce the amount of side effects in the according software. Simulation and debugging is more complicated, since there are two devices which have to be analyzed together. The same is applicable for *testability*. However, since the MCU-CPLD interface is simple and well defined, it should allow simulation and testing of both parts separately.

The *scalability* with respect to a number of frequency signals mostly depends on the chip area available on the CPLD.

Scalability with respect to the maximum frequency depends on the maximum clock frequencies of the CPLD. *Adaptability* with respect to additional functionality concerning signal generation depends mostly on the CPLD chip area available (see table 1). New requirements regarding the communication depend on the MCU. Changing, e.g. from RS232 to USB would only affect this part.

Time-to-market depends a lot on the expertise in the design team and additional support. The implementation involves the MCU part, the CPLD part, the communication part between the two devices and a board design integrating both devices. The costs consist of the costs for both hardware platforms, both development environments, a more complex printed circuit board (PCB), and protective circuits. The mounting space is probably larger than for a solution using only a single device.

3.4.4 Implementation 4: medium size FPGA

Since PLDs seem suitable for frequency generation, it would be interesting to integrate the whole functionality in a larger PLD, e.g. a medium size FPGA. For the RS232 communication, a soft core controller[2] is available which could be integrated into the design. An algorithm performing the CRC has to be integrated. The frequency generation is done as described in 3.4.3. The only control structure would be reading from the RS232 controller, perform the CRC and feed the message to the frequency generator. The *robustness* depends on the FPGA hardware, especially the *I/O capabilities*. To improve robustness protective circuits for the four frequency signals might be necessary. *Reliability* is better than in the previous version since no external communication path is present. Real-time simulation is available for this device. *Testability* is improved since all intermediate signal can be routed to FPGA pins or to internal test modules without influencing the main functionality. *Scalability*, with respect to number of signals and *adaptability* with respect to additional functionality concerning signal generation depends mostly on the FPGA chip area. Changing from RS232 to USB would mean replacing the RS232 soft core by an USB soft core (if available and small enough to fit in device). As in all cases, *time-to-market* does involve a lot the expertise of the design team. In contrast to 3.4.3, this approach only involves one hardware device and no interconnections are necessary. However, the realization of RS232 communication and CRC is probably faster on an MCU than on an PLD. The costs for the hardware platform is probably similar or slightly higher than in case of the second implementation, the costs of the development environment range from none to very high. As in section 3.4.2, the *mounting space* is determined by the package of the target hardware platform and the amount of external circuits needed for operation.

3.4.5 Hardware Selection

For the hardware selection, a survey concerning the suitability of the different hardware platforms is useful, e.g. in form of a table. This table should at least include functional and non-functional requirements which have been considered as important. For the simple example presented above this survey is done in table 3 with "++" representing *very good* to "-" representing *not suitable*. The functional requirements are fulfilled only by the implementations 2, 3 and 4. The platforms 2 and 4 seem to be suited best. Platform 4 seems to be very suitable if the focus is on adaptability and scalability, while platform 2 seems to be preferable if the focus is on fast development (assuming that the development team is familiar with all variations). As in this example the importance of fast development is rated higher than adaptability and scalability, platform 2 should be chosen.

[2]A core is a predesigned, preverified circuit block; a soft core consists of a synthesizable HDL (Hardware Description Language) description that can be retargeted to different devices (see e.g. [4]).

Table 3: Example: comparison of alternatives

requirement	1	2	3	4
performance	-	+	+	+
functional range	-	++	+	++
robustness	+	+	-	+/-
reliability	+/-	++	+	++
adaptability	-	+/-	+	+
scalability	-	+/-	+	++
testability	+/-	+	+/-	++
time-to-market	+/-	++	+/-	+/-
cost	++	+	+/-	+
mounting space	++	+	-	+

4. APPLICATION OF THE APPROACH IN EDUCATION

In accordance with [5, 9, 10] we believe that teaching embedded systems should include CPU and PLD based systems. In our opinion, one minor aspect is whether these different systems are taught together or one after another. However, it is of great importance to provide a comprehensive comparison of the alternatives available. Lab courses, as already mentioned, are well suited for this comparison. Since not all properties of the different hardware platforms could be taught in a lab course, they should be complemented by lecture contents. One possibility could base upon the approach presented in this paper. We will integrate this approach in one of our courses called *Introduction to Embedded Systems*. This course is intended for computer science students, typically in their 5th semester, and comprises the basic properties of embedded systems. Hardware platforms introduced in this course are MCUs, Programmable Logic Controllers (PLC) and FPGAs. In the corresponding exercises the students work with development boards in case of MCUs (Atmel ATmega16) and FPGAs (Xilinx Spartan3) and a simulation environment for PLCs (CoDeSys).

We plan an integration of our approach at the end of the course, as soon as students are familiar with the different hardware platforms. With this background the structure presented in this paper can be developed together with the students. First, system functionalities and qualities and their correlation to hardware properties could be discussed in a lecture. The second step, developing the more detailed tables, could be done by students in an exercise. Proposals could be discussed. It has to become clear that only general information could be included in theses tables. Information, e.g. specific for a certain MCU, could be included as an example, but usually has to be taken from the according data sheets. Later on, students could be provided with a set of tables including general information for embedded hardware platforms. This way, the approach presented could be used

as a comprehensive survey of the material taught.

This approach covers two issues important for embedded systems education. Firstly, it can give a survey of different hardware platforms existent in embedded systems and their influences concerning functional and non-functional requirements. This survey eases the understanding of similarities and differences of these hardware platforms. Secondly, the approach offers a systematic methodology for hardware platform selection based on functional and non-functional requirements. The approach itself is flexible to include additional properties of existent and new future devices and thus is not dependent on any device technology. This methodology should provide students with the skills necessary to use their knowledge in their own future selection processes in a systematic way.

5. DISCUSSION

The aim of this paper is to present a systematic approach for hardware platform design decisions which could be integrated into the embedded system education. The structures and tables presented are based on experiences gained and research done at our institute and are not claimed to be complete.

According to the high number of different and special functions in modern embedded hardware platforms and current changes in techniques and devices, it would be useful to provide a system in which "experts", as domain experts from academia and industry, and application engineers of the according hardware and software companies, could include their knowledge. A HTML-based web system in which participants could give feedback and propose additions to the structure and its contents could be one way to realize this integration of expert knowledge[3].

Due to limited space, the feasibility of the selection method has been illustrated with a comparatively simple application only. More complex applications would not change the selection method (hardware attribute tree, hardware properties influencing these hardware attributes, platform selection) itself, but would complicate the analysis process in case of some system requirements (e.g. reliability for platform A vs. platform B). The analysis of other hardware attributes are less affected by the application's complexity, as for example the marketability. This analysis is part of every selection process and does not represent any particular limitation to the approach presented in this paper.

We are going to integrate this approach into one of our next lectures, called *Introduction to Embedded Systems* which will allow us further evaluation of our approach.

6. CONCLUSIONS

In this paper we discussed the need for education in the field of hardware platform selection. Even if students learn about different hardware platforms used in embedded systems, a comprehensive comparison of these systems is often missing. To overcome this problem, a structured approach for teaching a systematic hardware platform selection process has been presented. In the first step of this approach, functional and non-functional requirements are mapped to hardware properties with the help of a graphical representation. This mapping is done on an abstract level which allows staying mostly hardware independent. In a second step, additional information concerning the hardware properties presented before, are provided in form of tables. These tables allow a separation between general information, specific information for CPU based hardware, and specific information for PLD based systems. Possibilities to integrate up-to-date information in this structure have been discussed. Our plans to integrate this approach in an embedded systems course have been presented. We believe that this two step approach provides a structure which could be readily integrated into embedded system education.

[3]planned web system:
www-i11.informatik.rwth-aachen.de/index.php?id=shps

7. REFERENCES

[1] M. Broy. Requirements engineering for embedded systems. In *Proceedings of FemSys'97*, 1997.

[2] M. Delvai and A. Steininger. Teaching hardware software codesign to software engineers. *1st International Workshop on Reconfigurable Computing Education*, 2006.

[3] B. Graaf, M. Lormans, and H. Toetenel. Embedded software engineering: The state of the practice. *IEEE Software*, volume 20:pages 61 – 69, 2003.

[4] R. K. Gupta and Y. Zorian. Introducing core-based system design. *IEEE Design & Test of Computers*, 1997.

[5] R. Hartenstein. The changing role of computer architecture education within cs curricula. Invited talk, Workshop on Computer Architecture Education (WCAE'04) at 31st International Symposium on Computer Architecture., 2004.

[6] R. Kazman, M. Klein, and P.Clements. Atam: Method for architecture evaluation (cmu/sei-2000-tr-004). Technical report, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

[7] J. Mylopoulos, L. Chung, and B. A. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *Software Engineering*, 18:483–497, 1992.

[8] F. Salewski, D. Wilking, and S. Kowalewski. Diverse hardware platforms in embedded systems lab courses: A way to teach the differences. In *First Workshop on Embedded System Education (WESE)*, volume 2. SIGBED Review, 2005.

[9] A. Sikora and R. Drechsler. *Software-Engineering und Hardware-Design*. Hanser Verlag, 2002.

[10] F. Vahid and T. Givargis. *Embedded System Design - A unified Hardware/Software Introduction*. Wiley, 2002.