Submission to RSP'2000

Title:	**Embedded System Architecture Design Based on Real-Time Emulation**	
Key Words	Methodologies for Hardware Prototyping and Software Prototyping, System Emulation, Prototyping of Real-Time Systems, The Role of FPGAs in System Prototyping, System Verification / Validation	
Authors:	Carsten Nitsch (principal author) Karlheinz Weiss Thorsten Steckstor FZI Embedded System Design Group	Prof. Dr. Wolfgang Rosenstiel
Address:	Forschungszentrum Informatik (FZI) at the University of Karlsruhe Haid-und-Neu-Str. 10-14 76131 Karlsruhe Germany Phone: ++49-721-9654-464 Fax: ++49-721-9654-409 E-mail: nitsch@fzi.de, weiss@fzi.de, stecki@fzi.de	FZI and University of Tübingen Lehrstuhl für Technische Informatik Am Sand 13 72076 Tübingen Germany Phone: ++49-7071-29-75482 Fax: ++49-7071-29-5062 E-mail: rosen@informatik.uni-tuebingen.de

Embedded System Architecture Design Based on Real-Time Emulation

Abstract

This paper presents a new approach to the design of embedded systems. Due to restrictions that state-of-the-art methodologies contain for hardware/software partitioning, we have developed an emulation based method using the facilities of reconfigurable hardware components, like Field Programmable Gate Arrays (FPGA). Our own emulation environment called the SPYDER tool set was used; it is best suited for the emulation of hardware designs for embedded systems.

1 Introduction

Most of today's existing technical applications are controlled by so-called embedded systems[1]. Many different application areas which demands their own specific embedded system architecture exist. Therefore, a common definition of embedded systems cannot find wide acceptance.[1]

In this domain, an embedded system architecture consists of an application-specific hardware part, which interacts with the environment. At the same time, an application specific software part runs on a microcontroller. In the last few years, rapid progress in microelectronic technology has reduced component costs, while simultaneously increasing the complexity of microcontrollers and application specific hardware.

Nevertheless, developers of embedded systems have to design low cost, high performance systems and reduce the time-to-market to a minimum. The most important taste a specification must complete is the partitioning of the system into 2 parts. The first part is the software which runs on a microcontroller. Powerful on-chip features, like data and instruction caches, programmable bus interfaces and higher clock frequencies, speed up performance significantly and simplify system design. These hardware fundamentals allow Real-time Operating Systems (RTOS) to be implemented, which leads to the rapid increase of total system performance and functional complexity. Nevertheless, if fast reaction times must be guaranteed, the software overhead due to task switching becomes a limiting performance factor and application-specific hardware must be implemented. This can be done by developing ASICs. Due to the decreasing life cycles of many high-end electronic products, there is a gap between the enormous development costs and limited reuse of an ASIC. In the last few years, so-called IP-Core components became more and more popular. They offer the possibility of reusing hardware components in the same way as software libraries. In order to create such IP-Core components, the system designer uses Field Programmable Gate Arrays instead of ASICs. The designer still must partition the system design into a hardware specific part and a microcontroller based part.

2 State of the Art

Basically two major design methodologies for embedded systems exist.

2.1 Hardware First Approach

The most commonly applied methodology in industry is based on a sequential design flow. This design-oriented approach is shown Figure 1:
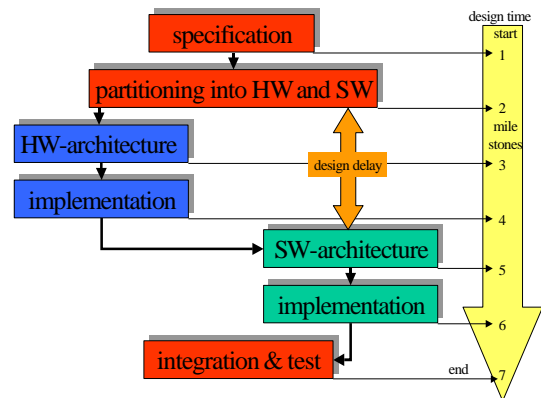

Figure 1. Design-Oriented Sequential Design Flow

The first step (milestone 1) of this approach is the specification of the embedded system, regarding functionality, power consumption, costs, etc. After completing this specification, a step called „partitioning" follows. The design will be separated into two parts:

- A hardware part, that deals with the functionality implemented in hardware add-on components like ASICs or IP cores.

- A software part, that deals with code running on a microcontroller, running alone or together with an real-time-operating system (RTOS)

The second step is mostly based on the experience and intuition of the system designer. After completing this step, the complete hardware architecture will be designed and implemented (milestones 3 and 4). After the target hardware is available, the software partitioning can be implemented.

The last step of this *sequential* methodology is the testing of the complete system, that means the evaluation of the behavior of all the hardware and software components.

Unfortunately developers can only verify the correctness of their hardware/software partitioning in this late development phase. If there are any uncorrectable errors, the design flow must restart from the beginning, which can result in enormous costs. For this reason, developers often use „well-known" components rather then new available circuits. They want to reduce the risk of design faults and to reuse existing know-how. This is especially important for the design of systems consisting of few, but highly complex components.

Another disadvantage of this approach is that it is not pos-

sible to start software development before the design and test of the hardware architecture has finished. Software developers have to wait until a bug-free hardware architecture is available. This time (and cost) intensive delay is graphically displayed in Figure 1 between milestone two and four.

Once again, the disadvantages of this methodology are: complete redesign in case of design faults, reduced degrees of freedom in selection of components (due to reuse of knowledge and experiences) and time delays. Nonetheless, the hardware-first approach is still a valueable approach to system design with low or medium complexity, because the initial step of partitioning is less time-consuming than in other approaches. For high-end embedded systems new methods are needed to recognize errors during an early phase of the design process.

2.2 Hardware / Software Co-Design

The first step in this approach focuses on a formal specification of a system design as shown in Figure 2. This specification does not focus on concrete hardware or software architectures, like special microcontrollers or IP-cores. Using several of the methods from mathematics and computer sciences, like petri-nets, data flow graphs, state machines and parallel programming languages; this methodology tries to build a complete description of the system's behavior. The result is a decomposition of the system's functional behavior, it takes the form of a set of components which implements parts of the global functionality. Due to the use of formal description methods, it is possible to find different alternatives to the implementation of these components.
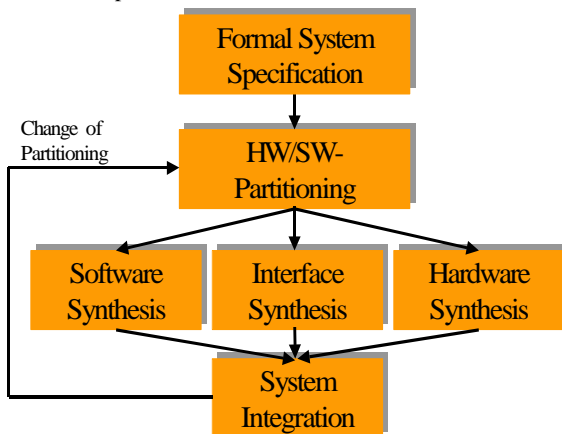

Figure 2. Hardware / Software Co-Design

The next step is a process called hardware/software partitioning. The functional components found in step one can be implemented either in hardware or in software. The goal of the partitioning process is an evaluation of these hardware/software alternatives. Depending on the properties of the functional parts, like time complexity of algorithms, the partitioning process tries to find the best of these alternatives. This evaluation process is based on different conditions, such as metric functions like complexity or the costs of implementation.

After a set of best alternatives is found, the next step is the implementation of the components. In Figure 2, these implementations are shown as hardware sythesis, software synthesis and interface synthesis. Hardware components can be implemented in languages like VHDL, software is coded using programming languages like Java, C or C++.

The last step is system integration. System integration puts

all hardware and software components together and evaluates if this composition complies with the system specification, done in step one. If not, the hardware/software partitioning process starts again.

An essential goal of today's research is to find and optimize algorithms for the evaluation of a partitioning. Using these algorithms, it is theoretically possible to implement hardware / software co-design as an automated process.

Due to the algorithm-based concept of hardware/software co-design there are many advantages to this approach. The system design can be verified and modified at an early stage in the design flow process. Nevertheless, there are some basic restrictions which apply to the use of this methodology:

- **Insufficient knowledge:** As described in this section, hardware/software codesign is based on the formal description of the system and a decomposition of its functionality. In order to commit to real applications, the system developer has to use *available* components, like IP-cores. Using this approach, it is necessary to describe the behavior and the attributes of these components completely. Due to the blackbox nature of IP-cores, this is not possible in all cases.

- **Degrees of freedom:** Another of the building blocks of hardware/software codesign is the unrestricted substitution of hardware components by software components and vice versa. For real applications, there are only a few degrees of freedom in regards to the microcontroller, but for ASIC or IP-core components, there is a much greater degree of freedom.This is due to the fact that there are many more IP-cores than microcontrollers which can be used for dedicated applications, available.

Due to the limitations that have been mentioned, the hardware-software co-design approach is not suitable for some design projects, like very complex systems used in automotive, aeroplane or space technologies.

2.3 Conclusion about these state of the art approaches

Both methods have their disadvantages. The hardware first approach does not allow verification of the system design at an early stage in the design flow process, the hardware/software co-design methodology is limited by insufficient knowledge about the internal behavior of hardware- or software IP-Cores and the restricted degrees of freedom in the choice of microcontroller components.

Both methodologies are unsuitable for developing embedded systems consisting of only a few, but nonetheless highly complex components.

Due to these analyses, we have developed another methodology which combines the advantages of the hardware-first-approach and the hardware/software co-design approach.

3 Emulation Based Methodology

Analyzing the hardware-first approach we have documented major advantage to this method. Developers using this design method focus on developing a prototype as soon as possible. This strategy complies with the major time-to-market constraints of today's high tech industry. To reduce the risk of design faults and cost intensive redesigns, system designers often use well known components instead of newly available

technologies.

Our design methodology tries to benefit from the advantages of rapid system design, without the disadvantages of the restrictions described in the previous section. The methodology can be described as a two-stage process:

- *Stage One - System design by evaluation:* The basic goal of this stage is the evaluation of components that can be used in the system design. In contrast to the classical hardware-first approach, this procedure is not restricted to known or already used hardware or software components. All potentially available components will be analyzed using criteria like functionality, technological complexity, or testability. The source of the criteria used can be data sheets, manuals, etc. The result of this stage is a set of components for potential use, together with a ranking of them.

- *Stage Two: Validation by Emulation:* Although stage one is based on functional and non-functional criteria, the knowledge and experience of the system designer still exerts a large influence on decisions. In order to avoid fatal design errors, stage two validates the decisions made in stage one. The basic methodology for this validation is system emulation. In contrast to other approaches like computer simulation, emulation can check „serious" problems, like real time behavior. It is highly essential to verify the criteria used in stage one, for example, the correctness of data sheet specifications.
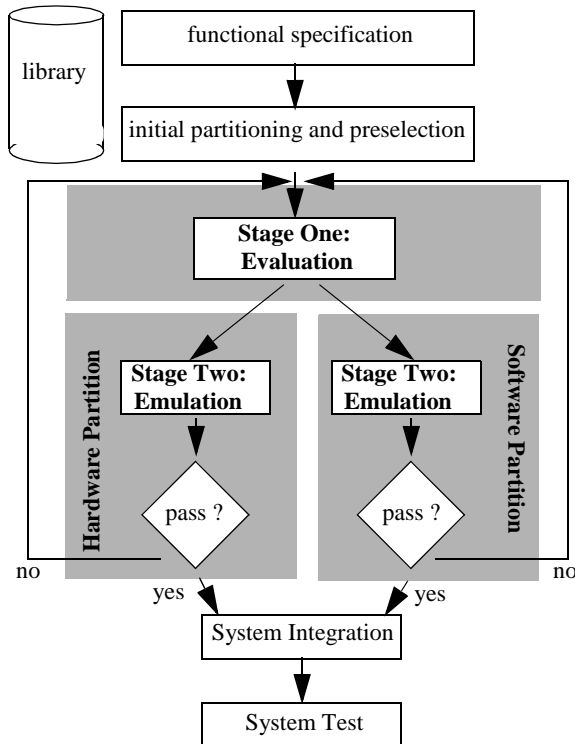

Figure 3. Emulation Based Methodology

Figure 3 gives an more detailed overview of our methodology. After the specification of the system design, the developer makes an initial hardware/software partitioning. The outcome is a set of hardware and software IP-Cores, the potential candidates that can be used to construct the system. The candidates can be selected from a library or another data base of informa-

tion. After these introductory steps, the first stage of our methodology follows. The evaluation and selection process focuses on a set of criteria, like testability. The output is a set of components which satisfy such special criteria in the best possible manner. Refer to [6] for a detailed description of this process.

After establishing the criteria, the already described „validation" stage follows. Only if a component passes this „test phase", it will be used in the final system design.

3.1 Stage One: Decision-making Criteria and Ranking

The previous chapter gave a short overview of the principles of our approach. The evaluation stage which was described is based on a process that puts together a ranking for components by focussing on special criteria. This chapter will explain how to define these criterias and which ranking will be used for selecting or throwing out components.

The most important component of an embedded system is the microcontroller. That is why there are only a few types of controllers available, but the choice of the microcontroller determines basics like the system bus, power supply voltages, etc. The first stage of our emulation-based design approach is aware of such choices, as Figure 4 shows.

The features of the microcontroller, especially performance determine what will be implemented as software. A system which is equipped with a high performance microprocessor can implement time-consuming functions, like MPEG-decoding software. If the microcontroller fails to complete this task, additional hardware must be added. Due to the high costs of ASIC design, the only possibility is to select the right components from a pool of available chips or IP-Cores.
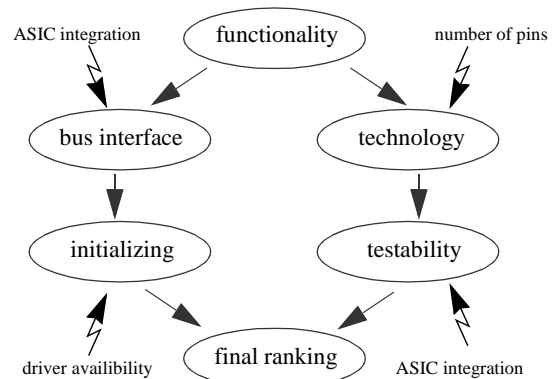

Figure 4. Decision Criteria

The next set of criteria for the selection of suitable components is the bus interface of the microcontroller. This criteria is more important than the other criteria shown in Figure 4, such as the initialization of a component. That is why the connection of a component to the microcontroller is essential for estimating the final costs and performance of an embedded system:

The best case is complete compatibility of the busses of the microcontroller and the component which are connected each other. Another possibility is that both bus interfaces are completely incompatible. To connect this type of hardware component to the microcontroller, highly complex bridges are necessary. The apparent disadvantages would be increased costs and significant communication delays between the micro-

controller an the newly added component. In regards to the contents of this chapter, it is possible to construct a ranking system to choose the most suitable component using the criteria bus-interface. The evaluation of the other criteria, like initialization, testability, the complexity of adding a component to a printed circuit board, etc., follows a analogical way. For detailed information refer to [6].

3.2 Stage Two: Validation by Emulation

For the emulation of hardware and software components, we have developed the SPYDER System. The basic idea of the SPYDER-System is to get a detailed view of the internal system behavior of complex embedded systems based on real-time emulation. In the past, these tools were used in different research projects published in [3][4][5]. The SPYDER System currently consists of two components:

- The SPYDER-VIRTEX-X2 Board for emulating application-specific hardware or testing IP-cores. This board covers the validation of the hardware partitioning of an embedded system design, (see Figure 3).

- The SPYDER-CORE-P2 Board is designed for emulating software components in a real-time environment. We have developed a Board Support Package for the VxWorks[1] RTOS. Due to the availability of this BSP, a variety of state of the art software IP-cores can be tested and benchmarked.

By referring back to figure 3, you can see that the hardware and the software partitioning be emulated and verified at the same time. This avoids time and cost-intensive delays between the phases of hardware design and software implementation in classical methodologies, like the hardware-first approach.

The next generation of boards that is currently being developed will combine these two platforms with additional internet based configuration features. This new platform will offer developers the chance to work together in a world wide distributed environment. Refer to section 4 for detailed information.
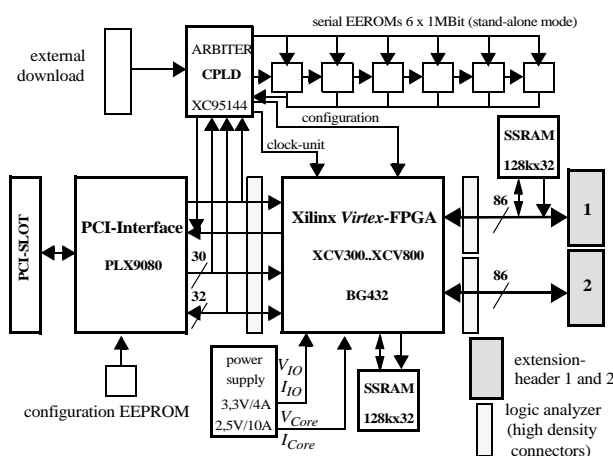
3.3 Emulation Platform SPYDER-VIRTEX-X2

Figure 5. Architecture of SPYDER-VIRTEX-X2

The basic component of the board is a *Virtex*[2] FPGA with

the package type BG432. Therefore, FPGA chips with a range of *XCV300* up to *XCV800* can be implemented. The architecture of SPYDER-VIRTEX-X2 is depicted in Figure 5.

The *Virtex* FPGA is closely coupled via a dedicated PCI-Interface-chip (*PLX9080*) to a PC. This feature enables both simple downloading bit images onto the *Virtex* chip and communication between the PC and the application operating on the *Virtex* FPGA via the PCI-bus, which provides a high performance bandwidth. The communication makes it possible to evaluate a running application, e.g., a dedicated IP-Core, before its integration into an embedded system. Using the PC with its entire periphery, (e.g. , display, hard disk, keyboard) instead of a specialized micro controller makes the evaluation process much easier.

Two powerful extension headers make it possible to connect the *Virtex* FPGA with further application specific hardware units, e.g., a micro controller and its core environment, as well as to assemble a complete embedded system architecture for emulation purposes. These ports are compatible with the other tools of the SPYDER-System mentioned above via backplane, which provides different micro-controller types. A further significant feature is the ability to connect all on-board signals via up to nine high density connectors to a logic analyzer. These connectors provide a powerful support during the debugging process. A power supply unit provides the *Virtex* FPGA with the necessary voltages: V_{Core} = 2.5V with a current of up to 10 A and V_{IO} with a current of up to 4 A. Two current measurement instruments can be connected inside the different current path systems for I_{Core} and I_{IO} to measure the power consumption. An arbiter controls the local side of the PCI-bus between PLX9080 and three different download modes, which can be summarized as follows:

- download via PCI-bus, set *Virtex* in slave mode

- download via external master, e.g. a micro controller-unit, set *Virtex* in slave mode

- download via serial EEPROMs, set *Virtex* in master mode, used for stand-alone mode

Additionally, two on board 128kx32 SSRAM-devices enable the emulation of applications, which need a large extension memory for, such things as graphic or large filter applications. For more information, refer to the corresponding user manual and data sheet in [2].
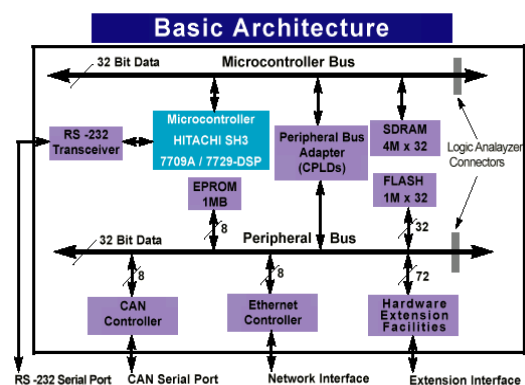
3.4 Emulation Platform SPYDER-CORE-P2

Figure 6. Spyder Core P2

The new high performance SPYDER-CORE-P2 board has been created for rapid, cost-effective development of software and hardware of the embedded system, mainly in the spheres of industrial automation, communication and automotive industries. The board is designed to run, test and evaluate application-specific software components, as well as software developed by the systems designer company itself or third party IP-cores. Referring to our design methodology shown in Figure 3, SPYDER-CORE-P2 covers the software portion of the design flow process.

Figure 6 shows the basic system architecture of the board. The core part is based on a novel 32-bit Hitachi-SH3 RISC micro controller with an optional on-chip Digital Signal Processing (DSP) module. Together with 1 MB of EPROM bootspace, 4 MB SDRAM and 1 MB of flash memory, the system offers all features to run state-of-the-art software components. Due to the availability of a VxWorks board support package, a wide range of RTOS based software can be tested.

SPYDER-CORE-P2 offers the most important interfaces to communicate with the surrounding environment. A standard serial interface, a CAN compliant controller and an ethernet 10Base 2 / 10BaseT interface are available. This ethernet feature allows the integration of the board in a fast ethernet-based development environment, for example the Tornado Toolkit.[1]

Two VG96 extension headers can be used to integrate additional hardware components. This feature allows the addition of application specific hardware, like additional memory, graphic controllers or other I/O facilities. All bus signals can be put through logic analyzer measurement by connectors of mictor type.

Together with SPYDER-VIRTEX or separately, SPYDER-CORE-P2 can be efficiently used in increasingly wider application areas. The board has been carefully optimized for high performance and low power consumption. It contains universal communication facilities which enable its usage in great variety of operational and development configurations. Supporting innovative design approaches and tools, SPYDER-CORE-P2 allows development, modification and the testing of new designs in shorter time frames, achieving high-quality characteristics.

4 Distributed Developing Environment: SPYDER-VIRTEX-X3

The next generation of boards called SPYDER-VIRTEX-X3 is currently under development. This system enhances the features of the SPYDER series. The most important new feature of the architecture is its scalability and the possibility to integrate the emulation system in a world-wide, internet-based, distributed environment.

4.1 Scaleability

In order to emulate an embedded system design, it is necessary to test several components and their communication with each other at the same time. Although the SPYDER System is highly qualified for validating hardware or IP-core components, there is a limitation given by the complexity of the FPGA chip used. In principle there are three ways to remove this barrier:

1. Use FPGAs with a higher gate density: This solution can be used for special designs needing a fixed number of gates for emulation. Due to increasing cost and tech-

nological and testing problems for ballgrid chips with hundreds of pins, this approach is not suitable as emulation environment, which could be put into common use.

2. Increase the ‚virtual' gate capacity of the FPGA by using an approach called „run-time-reconfiguration" (RTC). Run-time reconfiguration is a methodology focusing to a temporal partitioning of a hardware design. The result of this process is a set of time-exclusive functional components. Only one of these components will be active at the time t_0. The goal of RTC is to load design parts on demand. Using RTC allows the implementation of designs larger than the physical gate capacity of the FPGA, because not all parts are active at the same time. One important restriction of RTC is the necessity of the existence of time exclusive design components.

3. design a scalable emulation system

The new generation of our emulation system uses the third variant. Although the existing SPYDER-VIRTEX System can be scaled by connecting up to five boards by a backplane, there are some limitations regarding the configuration of the FPGAs. To make the FPGAs in-system-programmable, it is necessary to connect the boards with the PCI bus of a host PC. Due to the use of a PC there are restrictions in scalability (number of PCI slots). When dealing with automotive environments, aircraft etc., developers often have to test an embedded system without any additional equipment such as host PCs. They need a scalable and powerful emulation environment that also works in stand alone mode.

Considering this we have decided to develop a scalable emulation platform with a TCP/IP based interface for configuration and communication.
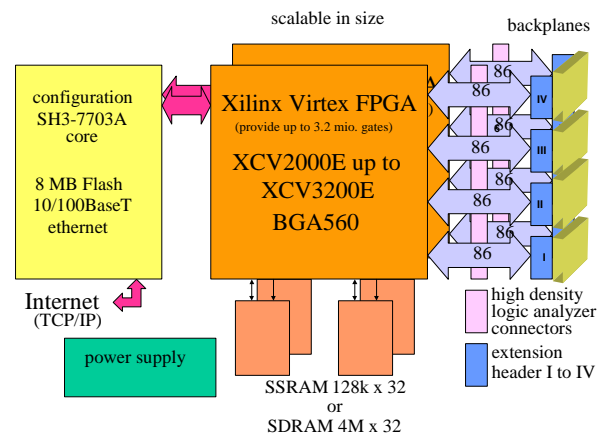

Figure 7. Basic Architecture of SPYDER-VIRTEX-X3

The SPYDER-VIRTEX-X3 acts as a master of a scalable environment as shown in Figure 7. Together with SPYDER-CORE or SPYDER-VIRTEX boards, the developer can use a powerful and flexible emulation environment for testing embedded system designs without a prototype. The configuration interface of SPYDER-VIRTEX-X3 is based on a Hitachi SH3 CPU running the real time operating system VxWorks. Due to the availability of a TCP/IP stack, this interface can use all TCP/IP based protocols, for example HTTP, FTP or proprietary protocols.

Figure 9 shows our FTP-based interface. The major parts

---

[1] Tornado is a trademark of WindRiver Systems

are a ftp server running on the real-time operating system VxWorks and a flash based DOS file system. Both the server and the flash drivers were developed by us to offer an easy-to-use reconfiguration environment. The software architecture is shown in Figure 8.
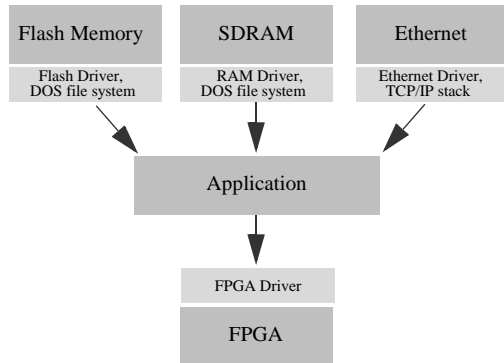

Figure 8. Software Architecture of SPYDER-VIRTEX-X3

This architecture supplies a set of hardware designs or IP-cores. These IP-cores are available as bit images, generated by software tools from VHDL-libraries or other file formats. These images can be selected and loaded onto the Virtex FPGA device. In Figure 8, there are basically two sources for tapping into the design files. The first one is the local file system, based on a persistent flash memory file system or a volatile, but quick, RAM disk. The second source is a world-wide network of external servers, available via the ethernet interface of the board and the TCP/IP stack of VxWorks. The modified FTP server is a frontend for managing all the sources and for integrating it in the environment of the developers host PC. Figure 9 shows the way this user interface works. Using a standard ftp client program, the developer can access the local file system (background window) of the emulation board as well as the design files stored on his developing host PC or on an external server (highlighted window). The most important feature of our ftp server is the so-called „special directory". These directories appear to the user as normal folders, but they are associated with device drivers.
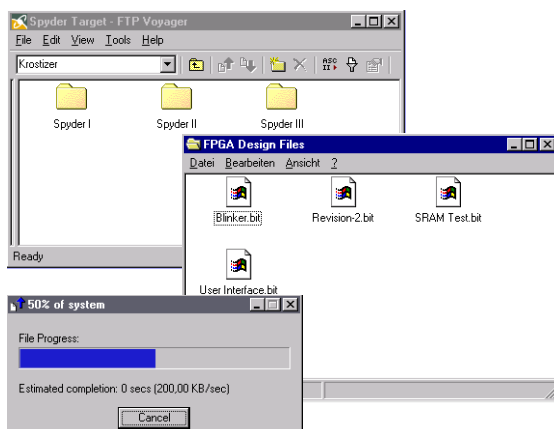

Figure 9. FTP based interface

By uploading a fpga image file in such a directory (i.e. , Spyder_I), the files are not stored on the target's flash file system, but will be redirected via a driver to the Virtex

FPGA device. This feature offers the ability to reconfigure FPGAs simply by using drag and drop. Due to the use of the FTP protocol, the developer can work with any host architecture he wants, such as windows based PCs as well as Unix workstations or Macintosh computers.

4.2 Distributed Emulation Environment

Due to the TCP/IP facilities of VxWorks, the emulation environment is not limited to a simple target/host development environment. Although the ftp interface still offers a comfortable and flexible working environment, there are much more powerful ways to benefit from the internet capabilities of SPYDER-VIRTEX-X3.

Due to the global structure of todays business, classical working environments become more and more obsolete. Imagine a virtual company developing and producing high-end electronic products. There are software developing offices in the United States, the hardware developers are Europeans and the production labs are located in Asia. The developer teams can work in different locations, some of them develop parts of the system like ASICs, other engineers have to integreate all parts of an embedded system.

The challange is to enable a „distributed office", that means offering the ability to work together in a simple fashion. Classical communication paths like email or the world-wide web are powerful, but sometimes restricted. These restrictions are based on the different know-how of the parties working together. Software developers have very good skills in debugging code of any kind, but they are not able to understand a complex hardware design in detail. Developing very complex systems, like an airplane or spacecraft, each developer only knows a part of the whole design in detail. Imagine the following situation as an illustration of this point. A hardware developer team designs a high-end ASIC with millions of transistors. Due to the complexity of the chip, there are more than one team, each developing a part of the chip. The system designers have to integrate the ASIC in an embedded system. Due to the complexity of the chip, it appears as a black box for these people. To fix a bug in the IP-Core design, normally the responsible ASIC team would have to change their design and to upgrade the ASIC. Of course they have to guarantee the consitence of the design. Due to the complexity of such an upgrade process, only the ASIC developers can do it. In pratice they design a new image file for a FPGA emulating the chip and the system designers have to load it onto a special FPGA to make it available on their system. Unfortunately, this process needs a lot of effort in the area of communication between the teams, so it is very time consuming.

A solution which avoids such problems is provided by the internet capabilities of the SYPDER-VIRTEX-X3 system. Using an configuration interface like the FTP server or maybe a JAVA-based software frontend, it is possible to upgrade the entire hardware of the system or parts of it via the internet. The IP-Core developers take care of the parts of the hardware they have developed, they can exchange it on the emulation system without any help from the system integrators. The system developers can focus their work on the whole embedded system.

5 Results

After the introduction of an emulation-based design methodology and the SPYDER tool set, which was developed by our

team, it is necessary to document our results and record our experiences with the system. The past three years were marked by the development of innovative embedded systems in the area of industrial automation, communication and automotive. This was done in cooperation with several companies in which these embedded systems were used for industrial applications.

5.1 Industrial Automation

A major project was done in cooperation with different industrial companies and led to the development of an Actuator Sensor Interface (ASI), a so-called ASI-Master. ASI is a new system which allows for the connection of up to 128 binary actuator and sensor devices with an appropriate control unit via a single bifilar cabel. An additional key feature of this work is the global access to the ASI-Master via the Internet, which leads to value-added services as described in [7]. Currently that project uses the RTOS *VxWorks.*

During the initial hw/sw partitioning, four tasks run on the RTOS; two are hard real-time tasks and two tasks have no real-time constraints.

1. The *Int_Service* task is a hard real-time constraint task and is responsible for the data exchange with the slaves. It generates the current process data image.

2. The *Control* task is also a hard real-time constraint task and uses the current process data image to calculate the control commands.

3. The server task has no real-time requirements and is responsible for data and command exchange via the Internet.

4. The embedded *C_Server* task also has no real-time requirements and transfers commands between a JAVA applet running on the calling client computer and the ASI-Master.

The basic idea of the SPYDER-System is to get a detailed view of the internal system behavior of complex embedded systems. In contrast to other approaches like computer simulation, emulation can check „serious" problems, like real time behavior. The emulation using the SPYDER System shows that an RTOS consumes a major part of the total execution resources, if the reaction times decreases down to the same delay as the task-switching times. In order to solve that bottleneck, the initial hw/sw partitioning based on an *Int_Service* routine in software must be changed and moved to hardware. For detailed information refer to [4].

5.2 Automotive Industries

In the last months, we worked closely with an automotive company developing different applications running a real-time operating System (RTOS). The company had used the hardware first method. After developing a complex prototype, the software developers started to code their applications.

While the automotive team was still developing their hardware, we had started to port the VxWorks RTOS using our SPYDER-CORE-P2 emulations system. This was possible and not difficult even without the availability of the customers own hardware. Benefiting from the debugging facilities of SPYDER-CORE-P2, VxWorks was ready to run within two weeks, before the hardware of our partner company was available.

Using the SPYDER emulation system, the behavior of the target's microcontroller architecture could be analyzed in detail by connecting it to a standard logic analyzer. It was possible to detect some tricky bugs and to fix them. Because the emulation system itself was tested, developers can focus on debugging their own applications, without regard for the problems of newly available prototypes.

To gain from the benefits of an emulation-based design methodology, our partner company has decieded to use this approach in the future by using the SPYDER tool set.

6 Summary

We started with the introduction of state-of-the-art methodologies for designing embedded systems, focussing on hardware-software partitioning. We have shown the basic restrictions of these classical approaches. Our solution to overcome these restriction is a new design methodology, which consists of two stages:

- preselection of available components
- validation by emulation

The major advantages of our methodology is a parallel design flow for hardware and software, rapid prototyping and the avoidance of dangerous design risks. We have developed an emulation system called SPYDER to use our approach with real system designs. The methodology and the SPYDER tool set are successfully applied in industrial OEM development projects. Our future work will focus on the internet integration of our emulation environment. The basic goal of our research activities is a world wide distributed development environment as introduced in section 4.2.

References

[1]   W. Wolf: *Hardware-Software Co-Design of Embedded Systems* Proceedings of the IEEE, Vol. 82, No.7, July 1994

[2]   I. Katchan, C. Oetker, T. Steckstor, K. Weiß: SPYDER-VIRTEX-X2 user manual, version 1.0, http://www.fzi.de/sim/spyder.html, september 1999.

[3]   K. Weiß, T. Steckstor, C. Nitsch, W. Rosenstiel: *Performance Analysis of Real-Time-Operation Systems by Emulation of an Embedded System.* 10th IEEE International Workshop on Rapid System Prototyping (RSP), Clearwater, Florida, USA, 1999.

[4]   K. Weiss, T. Steckstor, W. Rosenstiel: *Exploiting FPGA-Features during the Emulation of a Fast Reactive Embedded System.* ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), Monterey, CA, USA, 1999

[5]   K. Weiss, C. Oetker, I. Katchan, T. Steckstor, W. Rosenstiel: *Power Estimation Approach for SRAM-based FPGAs.* International Symposium on Field Programmable Gate Arrays (FPGA), USA 2000

[6]   K. Weiß: *Architekturentwurf und Emulation eingebetteter Systeme.* Ph-D. Thesis, University of Tübingen 15.Oktober 1999

[7]   A. Hergenhan, C. Weiler, K. Weiß, W. Rosenstiel: *Value-Added Services in the Industrial Automation.* ACoS'98, Lisabon Portugal, April 1998

射 频 和 天 线 设 计 培 训 课 程 推 荐

易迪拓培训(www.edatop.com)由数名来自于研发第一线的资深工程师发起成立，致力并专注于微波、射频、天线设计研发人才的培养；我们于 2006 年整合合并微波 EDA 网(www.mweda.com)，现已发展成为国内最大的微波射频和天线设计人才培养基地，成功推出多套微波射频以及天线设计经典培训课程和 ADS、HFSS 等专业软件使用培训课程，广受客户好评；并先后与人民邮电出版社、电子工业出版社合作出版了多本专业图书，帮助数万名工程师提升了专业技术能力。客户遍布中兴通讯、研通高频、埃威航电、国人通信等多家国内知名公司，以及台湾工业技术研究院、永业科技、全一电子等多家台湾地区企业。

易迪拓培训推荐课程列表：　http://www.edatop.com/peixun/tuijian/

射频工程师养成培训课程套装

该套装精选了射频专业基础培训课程、射频仿真设计培训课程和射频电路测量培训课程三个类别共 30 门视频培训课程和 3 本图书教材；旨在引领学员全面学习一个射频工程师需要熟悉、理解和掌握的专业知识和研发设计能力。通过套装的学习，能够让学员完全达到和胜任一个合格的射频工程师的要求…

课程网址：http://www.edatop.com/peixun/rfe/110.html

手机天线设计培训视频课程

该套课程全面讲授了当前手机天线相关设计技术，内容涵盖了早期的外置螺旋手机天线设计，最常用的几种手机内置天线类型——如 monopole 天线、PIFA 天线、Loop 天线和 FICA 天线的设计，以及当前高端智能手机中较常用的金属边框和全金属外壳手机天线的设计；通过该套课程的学习，可以帮助您快速、全面、系统地学习、了解和掌握各种类型的手机天线设计，以及天线及其匹配电路的设计和调试…

课程网址：　http://www.edatop.com/peixun/antenna/133.html

WiFi 和蓝牙天线设计培训课程

该套课程是李明洋老师应邀给惠普 (HP)公司工程师讲授的 3 天员工内训课程录像，课程内容是李明洋老师十多年工作经验积累和总结，主要讲解了 WiFi 天线设计、HFSS 天线设计软件的使用、匹配电路设计调试、矢量网络分析仪的使用操作、WiFi 射频电路和 PCB Layout 知识，以及 EMC 问题的分析解决思路等内容。对于正在从事射频设计和天线设计领域工作的您，绝对值得拥有和学习！…

课程网址：http://www.edatop.com/peixun/antenna/134.html

CST 学习培训课程套装

该培训套装由易迪拓培训联合微波 EDA 网共同推出，是最全面、系统、专业的 CST 微波工作室培训课程套装，所有课程都由经验丰富的专家授课，视频教学，可以帮助您从零开始，全面系统地学习 CST 微波工作的各项功能及其在微波射频、天线设计等领域的设计应用。且购买该套装，还可超值赠送 3 个月免费学习答疑…

课程网址：http://www.edatop.com/peixun/cst/24.html

HFSS 学习培训课程套装

该套课程套装包含了本站全部 HFSS 培训课程，是迄今国内最全面、最专业的 HFSS 培训教程套装，可以帮助您从零开始，全面深入学习 HFSS 的各项功能和在多个方面的工程应用。购买套装，更可超值赠送 3 个月免费学习答疑，随时解答您学习过程中遇到的棘手问题，让您的 HFSS 学习更加轻松顺畅…

课程网址：http://www.edatop.com/peixun/hfss/11.html

ADS 学习培训课程套装

该套装是迄今国内最全面、最权威的 ADS 培训教程，共包含 10 门 ADS 学习培训课程。课程是由具有多年 ADS 使用经验的微波射频与通信系统设计领域资深专家讲解，并多结合设计实例，由浅入深、详细而又全面地讲解了 ADS 在微波射频电路设计、通信系统设计和电磁仿真设计方面的内容。能让您在最短的时间内学会使用 ADS，迅速提升个人技术能力，把 ADS 真正应用到实际研发工作中去，成为 ADS 设计专家...

课程网址： http://www.edatop.com/peixun/ads/13.html

我们的课程优势：

※ 成立于 2004 年，10 多年丰富的行业经验，

※ 一直致力并专注于微波射频和天线设计工程师的培养，更了解该行业对人才的要求

※ 经验丰富的一线资深工程师讲授，结合实际工程案例，直观、实用、易学

联系我们：

※ 易迪拓培训官网：http://www.edatop.com

※ 微波 EDA 网：http://www.mweda.com

※ 官方淘宝店：http://shop36920890.taobao.com