

Emulation of a Fast Reactive Embedded System using a Real Time Operating System

Abstract

This paper presents the emulation of an embedded system with hard real time constraints and response times of about 220 μ s. We show that for such fast reactive systems, the software overhead of a real time operating system becomes a limiting factor. We analyze the influence of novel microcontroller features, e.g., different on-chip caches, which tend to accelerate execution, but make it less predictable. These investigations have been conducted using our own emulation environment called SPYDER-CORE-P1.

1 Introduction

Most of today's existing technical applications are controlled by so-called Embedded Systems¹. Many different application areas exist which demand their own specific embedded system architecture. Therefore, as described in [1], a common definition of embedded system does not exist.

The application presented in that paper is used in the area of industrial automation. In that 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 is running on a microcontroller. Especially the interaction with the system environment forces hard real time requirements.

In the last few years, the rapid progress in micro-electronic technology has reduced component costs while simultaneously making the introduction of the 32 bit embedded microcontroller [2] in a widespread area of embedded system design quicker. Additionally, powerful on-chip features like data and instruction caches, programmable bus interfaces and higher clock frequencies provide an enormous performance speed-up and simplify system design. These hardware fundamentals enable the implementation of a real time operating systems, which lead to the rapid increase in total

system performance and the complexity of the functionality.

Nevertheless, if fast reaction times of about 220 μ s must be guaranteed, the software overhead due to task switching becomes a limiting performance factor and the effect of the caches makes the performance analysis very difficult. These issues will be addressed in this paper, which is organized as follows:

In chapter 2, we give a short overview of the state of the art and our own previous work. Chapter 3 introduces the emulation environment SPYDER-CORE-P1, which was developed for the emulation of sophisticated embedded systems. Chapter 4 describes the benchmark application and the different software tasks, which are running under the control of the real time operating system VxWorks. Chapter 5 analyzes the performance effect of the different on-chip features provided by novel microcontrollers on overall performance and outlines the system bottlenecks. Chapter 6 describes the results of the investigation and chapter 7 summarizes the entire paper.

2 Previous Work

2.1 State of the Art

Hard real time requirements heavily influence the hardware/software partitioning in order to find a good solution with respect to performance and cost. The traditional method applied by most system designers today can be described as a two step solution. First, a printed circuit board containing all the selected chips is developed and after production, the necessary application software is written in the second step. The main disadvantage of this method is the lack of detailed knowledge about the internal embedded system behavior, which can only be determined very late in the design process. If system requirements are not met, corrections must be made late in the development process significantly increasing design time and total costs. Sometimes, the entire project must be cancelled due to time or budget restrictions.

The more scientific approach tries to describe the behavior of a system at a high level of abstraction. After the design entry step, some performance evalua-

¹This work was supported in part with funds of the Deutsche Forschungsgemeinschaft under reference number 3221040 within the priority program "Design and design methodology of embedded systems".

tion is done followed by co-synthesis techniques for software and hardware as described in [6][7]. The result is verified using one or more combined simulators. This approach suffers from the lack of an appropriate design language to describe all the aspects of an embedded system. A mixture of different languages must therefore be used. Each language addresses a separate part of the system, which makes design entry a time consuming and difficult task. Additionally, the complex interaction between the system and its environment must be modelled for simulation. That work is also very difficult and in some cases impossible without an unreasonable amount of resources, tools and design time.

If we compare the two design methods mentioned, we can say that using the traditional method leads to a system without no detailed knowledge of behavior during the design process and is mainly based on the experience of the designers to make it work. The scientific approach recognizes that fact, but suffers itself from a lack of appropriate tools to apply the method. That was the motivation for us to search for an applicable method for getting detailed information based on understandable measurements of the system during an early design stage. Furthermore, the method should be very close to the final target system and avoid estimations about the embedded system behavior to decrease the risk involved with the development.

An approach, which is able to provide these features is embedded system emulation. It offers the possibility to find the best hw/sw partitioning and the best usage of system resources early in the design process. Many different emulation environments exist. A few environments [8][9] are principally very flexible, but because of the high degree of flexibility are not suitable for an embedded system like the type of fast reactive embedded systems under consideration here. This is due to the great expansion of the resulting system, leading to the loss of real time capability. Therefore, we have developed the new emulation environment explained in chapter 3.

2.2 Our basic work

The past two years were marked by the development of innovative embedded systems in the area of industrial automation and communication. This was done in conjunction with different companies and these embedded systems are needed for industrial applications. This work serves as a basis for the current research work, which investigates more effective methods for embedded system design.

The design of an Asynchronous Transfer Mode (ATM) diagnostic monitor was characterized by the complex initialization process of Application Specific Integrated Circuits (ASIC). That work [3] was done using our first generation emulation environment. The main contribution to the current work documented in

this paper is the experience, that emulation is a powerful method to analyze the internal behavior of complex embedded systems and guiding the definition of the architecture of our second generation emulation environment called SPYDER-CORE-P1 [4], which will be described in chapter 3. It is used for the current presented work.

Another 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 up to 128 binary actuator and sensor devices with an appropriate control unit via a single bifilar cable. An additional key feature of this works is the global access to the ASI-Master via the Internet, which leads to value-added services as described in [5]. Currently that project uses the Real Time Operating System (RTOS) VxWorks, which is running on SPYDER-CORE-P1. It is used for the scheduling of different tasks. This application serves as an benchmark and is explained in chapter 4. That application example was selected due to three major characteristics:

1. Sophisticated software task architecture (RTOS)
2. Novel microcontroller architecture with caches
3. Fast reaction times to external events

These characteristics make up a complex internal system behavior, which is therefore suitable to demonstrate the benefits of the emulation method in order to give answers to three important questions which arises from a system designers point of view:

1. What is the optimum clock speed in order to meet all real time constraints and to prevent an oversized or an undersized system?
2. How much computation performance is consumed by the RTOS in a fast reactive system?
3. How great is the performance enhancement of the caches?
4. What can be done with the cache enhancement?

This paper shows, that these questions are not limited to that specific application, but are general questions about many applications in the mentioned area.

3 Emulation platform SPYDER-CORE-P1

The current version of the SPYDER emulation environment consists of two boards. SPYDER-ASIC-X1 [10] is one part of the tool set and mainly addresses the emulation of large VHDL-based ASICs designs on a FPGA. SPYDER-CORE-P1 is the second part of the tool set and is used for the emulation of the entire embedded system architecture. Both boards are com-

patible and can easily be connected to each other via a backplane. For the work described here, only SPYDER-CORE-P1 was used, thus only this board is described in detail (see figure 1). The SPYDER-CORE-P1 environment provides all the key components needed for embedded systems in the area of industrial automation within a flexible, but still compact architecture, which guarantees high clock speeds. This enables a real time emulation, which is very close to the final target system.

A 32 bit RISC embedded PowerPC 403GA/GCX microcontroller [2] is used. It provides the following three advantages:

1. It is available in a wide clock frequency range. This enables the emulation of a large area of applications. An analysis step can achieve the right values to satisfy all real time requirements and prevent an oversized or undersized system.
2. The microcontroller architecture provides different types of on-chip caches (instruction and data, each having different sizes). The analysis of performance improvements with respect to different cache operations in fast real time systems, which are running under control of a RTOS, is an interesting research topic.
3. A flexible programmable bus interface provides a direct interface to most peripheral devices. This enables the simple implementation of a low or high end system without additional glue logic.

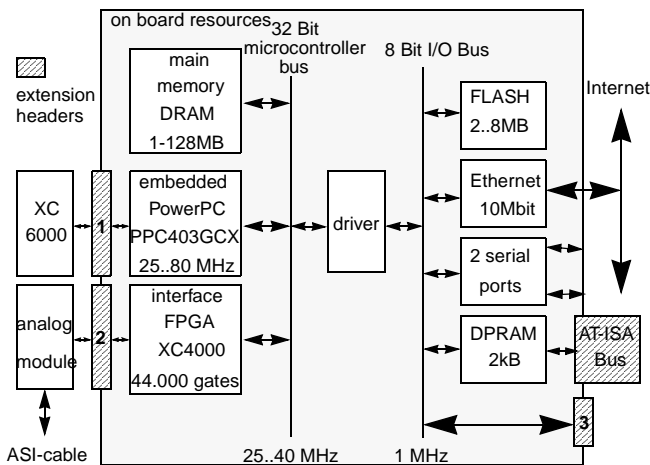


Figure 1. Architecture of SPYDER-CORE-P1

The high-speed microcontroller bus is connected to the main memory and the interface FPGA. It can be used to connect the entire microcontroller bus to external devices via the extension header 2 or it can be used to emulate additional application specific interface

hardware.

Each microcontroller bus signal is available on the extension header 1. They provide the means to connect a co-processor, e.g., a reconfigurable XC6000 FPGA, to the microcontroller very closely or to connect debugging equipment, e.g., logic analyzer, to the SPYDER-CORE-P1 emulation environment.

A driver device implements a buffer between the high-speed 32 bit wide microcontroller bus and the slower 8 bit I/O bus. It connects some frequently used communication and memory devices to the microcontroller.

4 Internet controlled ASI-Master

The Actuator Sensor Interface (ASI) connects up to 32 ASI slave chips via a single, bifilar cable with a ASI master unit (see figure 2). That master unit calls in a polling cycle each connected ASI slave with its own address (bits A4..A0), followed by the output data image (bits I4..I0). The ASI slaves respond, if the address in the master call matches with their own address and transfer the input data image back to the master. Each slave is able to connect to up to four binary sensors (4I) and up to four (4O) actuators. The serial master call protocol must be modulated on the ASI cable using a dedicated analog module (see figure 1). An ASI slave is available as a single chip solution which has the same capability on-chip. Additionally, an ASI power supply unit provides 24 DC voltage for the slaves via the same cable.

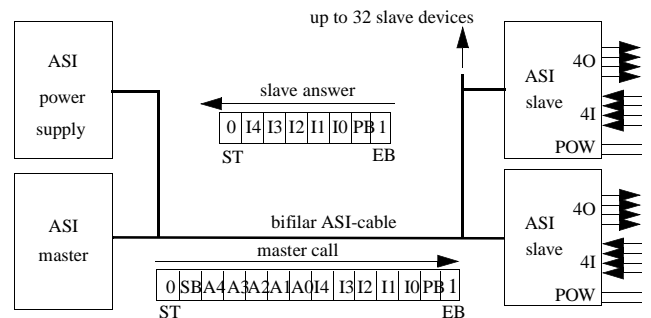


Figure 2. ASI Communication System

The ASI master is emulated on SPYDER-CORE-P1 very close to the final target system without any major changes. The hardware connection between the microcontroller and the analog module (two wires, serial in and out) is implemented as a dedicated ASI hardware in the interface FPGA XC4000 (see figure 3). The microcontroller writes the output data image to the register file and the ASI UART performs a parallel to serial conversion with manchester encoding. The digital serial data output is modulated to the ASI cable by the analog module. The receive path operates analogously to the send path.

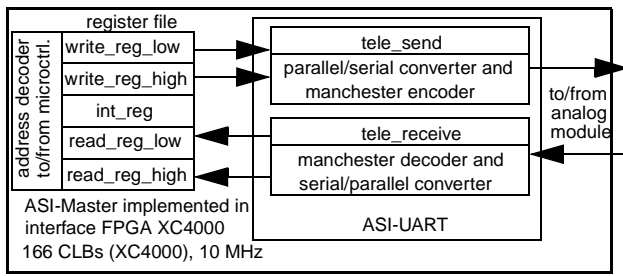


Figure 3. ASI Interface Hardware

The Real Time Operating System (RTOS) VxWorks [11] runs on the SPYDER-CORE-P1. It provides a TCP/IP stack for communication via ethernet. This interface is used as a link to the VxWorks development environment running on a host PC. Additionally, the ASI application uses that interface as a link to the Internet.

Four tasks run on the RTOS, two are hard real time tasks and two tasks have no real time constraints (see figure 4).

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 http server task also has no real time requirements and transfers JAVA applets to the calling client computer.

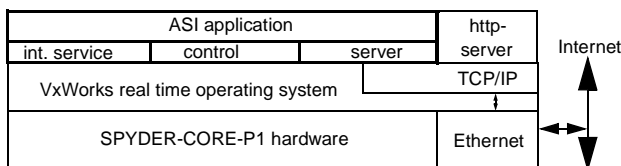


Figure 4. Software Architecture

For a more detailed description, refer to [5].

5 Embedded System Performance Analysis

Figure 5 shows a timing diagram of the hard real time tasks, measured with a logic analyzer. The int-signal shows the interrupt communication between the ASI specific hardware implemented in the interface FPGA and the microcontroller. The int-service and control tasks mark their beginning and end states by

accessing a simple I/O device (I/O-signal), which is recorded by the logic analyzer. The ASI standard defines the time delay between to serial bits on the ASI cable during the master call and slave answer with $6\mu\text{s}$. Together with master and slave break times, the microcontroller must exchange data with a slave every $220\mu\text{s}$. Therefore, this value is an ASI specific real-time critical constant. During this time, the microcontroller must execute the following sequence, interrupt reaction, interrupt service, task change, run control task and take a semaphore (semTake). To determine the minimum clock speed, all caches must be disabled due to the fact, that this sequence is a hard real time constraint and a worst case analysis has to be done. Figure 5 shows, if the worst case scenario occurs, it will still run successfully with a minimum clock speed of 33MHz or more. If the clock speed decreases, the next interruption (int = low) happens before the semTake currently being executed has finished. This means, the real time requirement can no longer be satisfied.

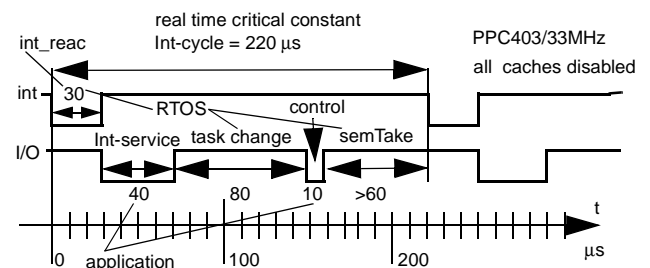


Figure 5. Execution Times

Figure 6 shows the clock frequency range from 25 up to 80 MHz on the x-axis. The y-axis shows the ratio of the value of the real time required for the execution of the mentioned task sequence (see figure 5) and the real-time critical constant. The upper curve depicts the case without all on-chip caches, which must be used to determine the worst case scenario for hard real time conditions. If the PPC403 operates at 33 MHz and the worst case scenario occurs (total cache misses and flushes), it is still able to guarantee the real time requirement. If the clock frequency decreases, the y-value increases above one. That means, the microcontroller performance is undersized. If the frequency is greater than 33MHz, the system is oversized.

The lower curve shows the behavior with enabled D & I caches. This is the normal operation state. The optimal working frequency (33MHz) provide an average gain of about 40% in execution performance. This 40% gain of the microcontroller execution performance can not be guaranteed in all cases, it is possible that lesser percentage occurs, although this performance gain usually can be expected. Therefore, these resources can only be used for no real-time tasks. In the range of 40MHz up to 80MHz, a further signifi-

cant performance gain can be verified. This is due to the fact that the 403GCX type in that speed range used has eight times larger caches than the 403GA type.

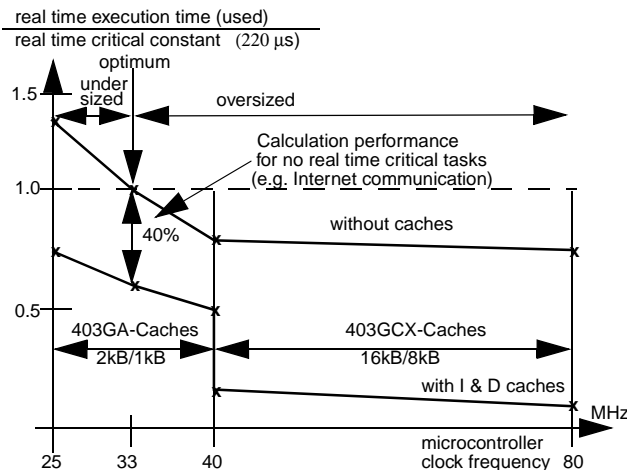


Figure 6. System Execution Resources

6 Results

The results give answers to the questions formulated in chapter 2.2 and can be summarized as follows:

1. The emulation gives a detailed view account of internal behavior in an early design stage and shows the optimal working frequency at 33MHz, which avoids an oversized or undersized system. The emulation reflects the final target system very closely without any change.
2. State of the art RTOS achieves task switching times of about 80 μ s and interrupt reaction times of 27 μ s. As shown in figure 5, the total time the RTOS uses is 170 μ s and the total time for the application is 50 μ s. Therefore, 77% of the total execution time (220 μ s) is consumed by the RTOS. That means, if the external environment of a fast reactive system forces interactions (here 220 μ s) to the same degree of magnitude as the task switching time (here 80 μ s), the software overhead of the RTOS becomes a limiting factor for the total embedded system performance.
3. Caches, which are enabled, improve execution performance and provide a gain of 40%.
4. If a system requires hard real time conditions, this is done without cache enhancements. Such enhancements can only be used for non-real-time dependent system services, e.g., network communication via the Internet.

7 Summary

System performance is influenced by many indeterministic parameters. In order to meet all real time requirements and exploit all available system resources, a detailed view of the internal behavior is necessary. Therefore, this paper presents the emulation of a fast reactive embedded system, running under control of a RTOS, as a powerful method to solve the problems mentioned and shorten the design time and risks.

In this paper, a novel emulation environment called SPYDER was introduced, which emulates very close to the final target system. A benchmark design with an industrial background was used to demonstrate the method. The optimal working frequency lies at 33MHz and the emulation shows, that a RTOS can consume up to 77% of the total execution resources, if the reaction times decreases down to the same delay as the task switching times. Additionally, the effect of on-chip caches is determined with a 40% performance gain, which can be used for such things as non-real-time critical network communication. This enables the total exploitation of all system resources and leads to value-added system services without higher clock speeds.

References:

- [1] W. Wolf: *Hardware-Software Co-Design of Embedded Systems*. Proceedings of the IEEE, Vol. 82, No.7, July 1994.
- [2] *PPC403GA/GCX Embedded Controller - User Manual*. IBM Corporation 1997
- [3] Karlheinz Weiß, Ronny Kistner, W. Rosenstiel: *Analysis of the XC6000 Architecture for Embedded System Design*. Field-Programmable Custom Computing Machines (FCCM), Napa Valley CA, April 1998,
- [4] Karlheinz Weiss, Thorsten Steckstor, Carsten Oetker, Ronny Kistner: *Data Sheet and User Manuel SPYDER-CORE-P1*. <http://www.fzi.de/weiss.html>, FZI & University Tübingen 1998
- [5] A. Hergenhan, Christoph Weiler, Karlheinz Weiß, Wolfgang Rosenstiel: *Value-Added Services in the Industrial Automation*. ACoS'98, Lisabon Portugal, April 1998
- [6] Sanjaya Kumar, James H. Aylor, Barry W. Johnson, Wm. A. Wulf: *The Codesign of Embedded Systems*. Kluwer Academic Publishers, 1996
- [7] Rajesh Kumar Gupta: *Co-Synthesis of Hardware and Software for Digital Embedded Systems*. Kluwer Academic Publishers, 1995
- [8] S. Hauck, G. Borriello, C. Ebeling: *Springbok: A Rapid-Prototyping System for Board-Level Designs*. ACM/SIGDA 2nd. International Workshop an Field-Programmable Gate Arrays, Berkley, Feb. 1994.
- [9] G. Koch, U. Kebschull, W. Rosenstiel: *A Prototyping Environment for Hardware/Software Codesign in the COBRA Project*. Third International Workshop on Hardware/Software Codesign, 1994.
- [10] Karlheinz Weiss, Thorsten Steckstor, Carsten Oetker, Ronny Kistner: *Data Sheet and User Manuel SPYDER-ASIC-X1*. <http://www.fzi.de/weiss.html>, FZI & University Tübingen 1998
- [11] *VxWorks Reference Manual and Programmer's Guide*. WindRiver Systems, Edition 1, 1997