Embedded Operating System for MicroBlaze

M. Šimek

This paper presents a work in progress on experiments with embedded operating systems for the MicroBlaze processor. Modern embedded systems based on a configurable platform incorporating a similar processor core are gaining importance with the ongoing effort to minimize cost and development time. After an overview of the configurable platform based on this processor core, we devote our attention to uClinux OS. This OS has been successfully ported for the MicroBlaze processor, and we present our current experience with it. At the end of the paper we discuss several possible booting strategies and recommend further development of U-BOOT.

Keywords: Embedded systems, operating systems, configurable HW, MicroBlaze.

1 Introduction

Customers have quickly become accustomed to the high standard of today's information technology, and their demands continue to grow. For this reason, modern embedded systems frequently resemble small computer systems. Just a few examples are PDAs, smart and cell phones, set top boxes, network and telecom systems. Systems like these require some degree of configurability, so that new functionality may be added at any time, even after production. This process is usually known as a firmware update. Platforms based on FPGAs have partially managed to address these issues.

Now is the right time to consider another degree of flexibility – it is time to consider the need to use an embedded operating system (OS). One of the key advantages of any OS is easy software integration. In the case of embedded systems, this allows an increasing amount of their functionality to be moved to software, instead of designing it as a pure hardware solution. The flexibility of software solutions and the lower design times help on the one hand to reduce the economically crucial time-to-market factor, and on the other hand to tune the most time consuming tasks in hardware.

Hardware acceleration and reconfiguration for time-consuming and computationally heavy applications is much more applicable in a fully configurable embedded system with the help of some OS.

2 Motivation

Aiming to follow modern trends and to evaluate for ourselves the real advantages of an embedded OS, we started a project on experiments with some OS. Because of our interest in configurable hardware, we chose the MicroBlaze processor [1] as the basis of the project.

First, there was a very important and difficult decision on which operating system would be the best for our platforms. The fundamental criteria were source code availability and support for a wide range of hardware devices (drivers) and standard software components (file systems, networking, etc.). Easy adaptability to a changing hardware platform was also crucial for the configurable systems that we considered.

Among the OSes that supported MicroBlaze we finally chose a Linux based system. Firstly, it matched up to our requirements, and secondly our familiarity with Linux systems was a decisive factor. Specifically, we used a distribution called uClinux [2, 3] (pronounced "you see linux").

3 Target system

3.1 MicroBlaze processor

MicroBlaze is a 32-bit embedded soft-core processor with a reduced instruction set computer (RISC) architecture. It is highly configurable and specifically optimized for synthesis into Xilinx field programmable gate arrays (FPGAs).

The MicroBlaze configurability enables embedded developers to tune its performance to match the requirements of target applications. For example, MicroBlaze may be configured to use a hardware multiplier or a dedicated barrel shifter. The current version even features an optional floating-point unit that can accelerate system performance by as much as 120 times over software emulation. Xilinx claims that the core can operate at frequencies up to 200 MHz.

3.2 Hardware platforms

Currently, we are using two development kits by Xilinx – ML401 and ML310 [1]. The former is based on Virtex-4 FPGA, while the latter uses Virtex-II Pro. Both platforms offer a wide range of industry standard peripherals – e.g. Ether-

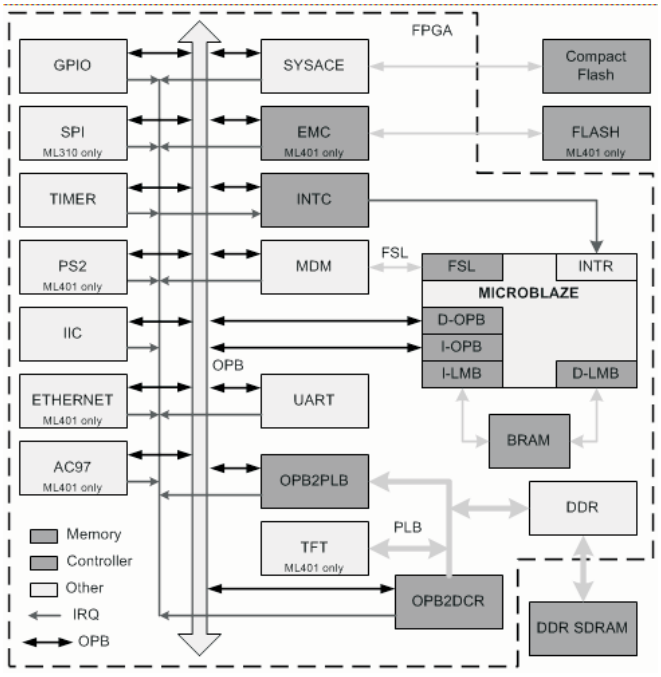

Fig. 1: Configurable system architecture

net controller, compact flash card controller, USB controller, AC97 audio codec, etc. All of these may optionally connect to MicroBlaze through configurable interfaces realized in FPGA. An example of what the final system may look like is shown in Fig. 1.

When using embedded OS, the Ethernet interface becomes of high importance, either as a standardized communication interface or for better support for applications development (i.e. debugging). The networking interface is famous especially for Unix based operating systems, among which uClinux may be classified. In addition, we use Ethernet for downloading new drivers to a board via FTP protocol.

3.3 uClinux operating system

uClinux has features similar to those of standard Linux, but its advantage is the optimization for embedded devices and applications. It is especially optimized to minimize the size of the code necessary for both applications and kernel. Unlike "standard" Linux, uClinux may be used even for embedded systems that have a main memory size as big as an L2 cache in an ordinary personal computer.

Like any other Linux system, uClinux is composed of a kernel and a distribution. Its kernel is derived from a standard Linux kernel v 2.0 with memory management left out. Today's kernel version is 2.4.32-uc0 and version 2.6 is planned for the near future. The kernel supports many processor families, such as Alpha, ARM, Blackfin, i386, m68k, MicroBlaze, MIPS, PPC, SH, SPARC, etc.

The main purpose of a distribution is in the first step creating the root file system and adding applications. The type of root file system is elective for almost any available storage device. We use two types of root file systems – ROMFS (ROM File System) and CRAMFS (Compressed ROM File System) [2]. CRAMFS is 20 % smaller than ROMFS. It is possible to use a NFS (Network File System), which wasn't fully tested yet.

The distribution extends the kernel for a number of programs. These include core applications (init, agetty, cron, at), flash tools (netflash, mtd-utils), file system applications (flatfsd), network applications (dhcpd, ftpd, inetd, ping, telnetd, thttpd, tftp, ifconfig, route), miscellaneous applications (cat, cmp, cp, ln ls, mkdir, mv, rm, ps), MicroWindows (still not tested), etc. A very useful tool is the Busybox package [4], which contains programs for managing kernel modules (e.g. mount, umount, insmod, lsmod, rmmod, modprobe).

4 Project status

4.1 Completed work

Operating systems such as Linux are very extensive, and it is hardly possible for an individual to fully comprehend them. However, a detailed knowledge of system internals is crucial in the embedded field, where each system is specific in some way.

Therefore, one of the most important objectives of the project was to gain enough experience with uClinux to be able to deploy the system on any HW platform. It was necessary to understand the kernel source tree structure, to go through kernel sources and discover all kinds of dependencies. Alongside this tedious work, kernel configuration and building seemed easy – though some problems had to be solved, too.

In this phase of the project, we took advantage of the fact that a working uClinux demo was available for the ML401 platform [3]. This was especially helpful at the beginning for purposes of testing the kernel and distribution configuration. With increasing experience, we used our own derived variants of this reference platform. Finally, to demonstrate our full mastery, we successfully ported uClinux to the ML310 development kit [5].

4.2 Full-platform support

uClinux is well-known for offering broad support for various hardware devices. However, one cannot expect this to be true for specifically designed components, such as configurable systems built within FPGAs.

Therefore, to get full support for our HW platforms, we need to write our own device drivers. A VGA controller driver is currently under development, and for the future, a driver for the USB controller is planned.

4.3 Booting strategy

So far, we have been using a simple boot loader created as a part of the project. Its only purpose is to initialize the necessary peripherals (serial line, LCD display), perform a memory test and, if successful, copy the kernel binary image into RAM and execute it. This procedure is illustrated in Fig. 2.

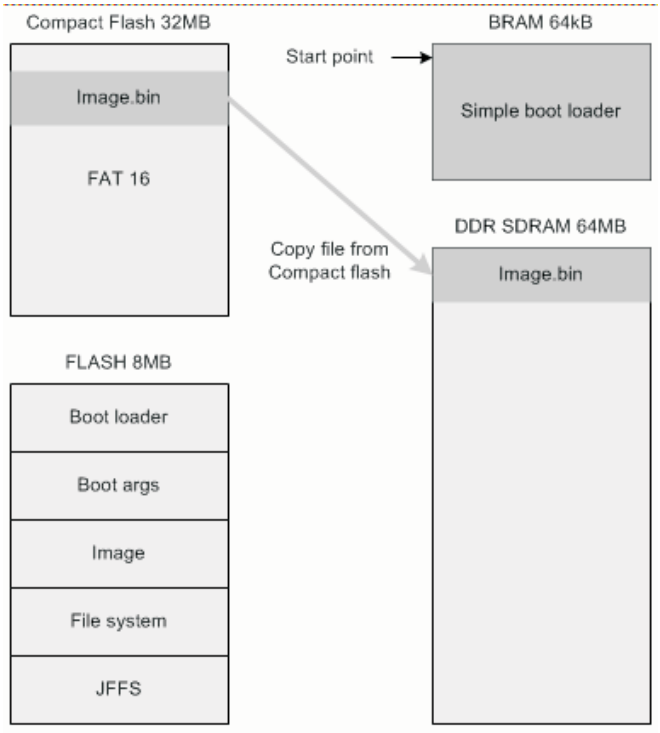

Fig. 2: Simple booting strategy

Although this simple boot loader works satisfactorily, it does not fully cover our needs. For higher configurability and better debugging means, it would be better to have a more sophisticated boot loader. This might for example allow us to pass boot arguments to the uClinux kernel, or choose between several pre-configured kernel images. With support from the Ethernet interface, it would even enable remote kernel updates. All these features and even more can be found within U-BOOT [6] (Das U-BOOT – universal boot loader).

U-BOOT is an open source project and it has been designed mainly for high flexibility. Its support for many processor families is also advantageous – e.g. ARM, i386, MIPS, Nios, PowerPC, Xscale, etc.

MicroBlaze is also supported, but the recent U-BOOT port for this processor has very limited capabilities. It implements only a serial line interface and allows us to work with RAM memory - listing, writing, modifying [7]. Therefore, our recent plan is to focus on implementing the remaining functions, e.g. remote file download, access to flash memory, support for boot arguments, etc.

With help of U-BOOT, we can easily implement our sophisticated booting strategy (see Fig. 3). This enables us to have three kinds of root file systems – standard ROMFS in RAM, a read/write file system on an external storage device partition, and finally an alternative JFFS system (The Journaled Flash File System) stored in a Flash memory. It would then be possible to access both Flash and external storage memory, and to choose a right root file system.

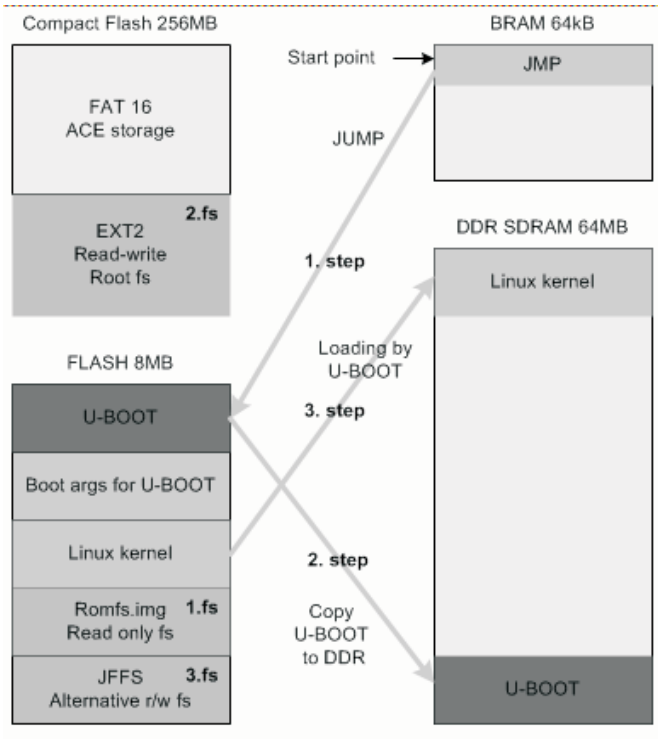

Fig. 3: Sophisticated booting strategy

5 Summary

We give an overview of a project in progress that concentrates on adapting a uClinux embedded OS for the MicroBlaze soft-core processor. We have given a brief overview of both uClinux and MicroBlaze. We have also presented some of the HW platforms on which we have been carrying out our experiments.

Although our position was simplified by the fact that a uClinux port for MicroBlaze already existed, it was still quite a tedious task to gain sufficient mastery of the system. However, this work has paid off because we are now able to adapt uClinux according to our needs. This ability is essential for configurable embedded systems, which are our main concern.

We are currently developing device drivers for unsupported peripherals and for designing a sophisticated strategy offering a convenient system boot procedure with unique debug properties. Our final objective is to provide a preconfigured uClinux distribution for the MicroBlaze processor that will be complete and flexible in support of our development platforms. Such a uClinux package would form the basis for further projects, which might then concentrate on some specific problems rather than dealing with the entire complexity of the operating system.

6 Acknowledgments

I would like to thank Tomáš Brabec for his help in completing of this paper.

References

[1] Xilinx: *The Programmable Logic Company*, [online] http://www.xilinx.com, 2006.

[2] Dionne, D. J., Albanowski, K., Durrant, M.: *uClinux – Embedded Linux/Microcontroller Project*, web page available at http://www.uclinux.org, 2006.

[3] Williams, J.: *MicroBlaze uClinux Project Home Page*, [online] http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux, 2006.

[4] Landley, R.: *Busybox: The Swiss Army Knife of Embedded Linux*, [online] http://www.busybox.net, 2006.

[5] Šimek, M.: *Embedded Operating Systems for Microblaze*, [online] http://cs.felk.cvut.cz/~simekm2/uclinux, 2006.

[6] U-BOOT: *Das U-Boot – Universal Bootloader*, [online] http://sourceforge.net/projects/u-boot.

[7] Shoji Yasushi: *SUZAKU: Series of Embedded Devices Based on the Combination of FPGA and Linux*, [online] http://suzaku-en.atmark-techno.com.

Michal Šimek
e-mail: simekm2@fel.cvut.cz

Dept. of Computer Science and Engineering

Czech Technical University in Prague
Faculty of Electrical Engineering
Karlovo náměstí 13
121 35 Praha 2, Czech Republic