Implementation of a low cost embedded system using the Leon2 processor

H. Guzmán Miranda, J. Tombs, M. A. Aguirre Echánove

Abstract—**Over the last few years, embedded systems have experienced an inmense growth both in computing power and scope of their possible applications. If embedded systems are to reach their full application domain, their total cost, both development and system cost, must be reduced as much as possible. More often than not, the cost of development software licenses, IP cores and embedded operating system licenses burden small and medium business and development groups. Usally the purchase of expensive development kits is mandatory in order to create a working environment with which to develop the embedded product. The present article proposes a low-cost solution based on Linux running on the Leon 2 processor, which is implemented on a low profile FPGA. This solution allows the consideration of systems that, because of their low cost, open source and easy expansion capabilities, present themselves as an interesting alternative for the implementation of embedded systems. To demonstrate the feasibility of this low cost approach, a minimal system has been implemented on purpose on a discontinued development board which is (at the moment of the implementation) at least five years old.**

Index Terms— **Embedded Linux, Embedded systems, FPGA, Low cost, SoC**

I. Introduction

WHEN designing an embedded system, it is necessary to reach a compromise between system cost, capacity and future expansion capabilities. Nowadays, there are many options from which the designer can choose at the time of conception of a system of this kind, among them the designer must select a specific microprocessor, and also decide if programmable logic will be used or not. Modern systems on chip allow to implement, on a single device, complex systems that otherwise would need of several on-board chips. By using an FPGA instead of a commercial microprocessor, this FPGA serves a double function: it is suitable both for supporting final designs and for prototyping ASICs.

Also, in many cases it is desirable to have the capability to add new functionalities to the system, for example to have the option of extending its features after the system has been deployed, or for the convenience of using the same hardware platform for developing solutions for different applications. A single hardware platform can be adapted to different applications, with the possibility of being optimized for the particular problem which needs to be solved.

There are different commercial packages available for System on Chip design, like Xilinx EDK or Altera Nios II IDE, which lighten the workload of the embedded system designer, providing proprietary softcores and the tools needed for implementing them in the manufacturer's FPGAs. These commercial solutions present some limitations, the most important being that the implemented softcores are dependant on the manufacturer's specific hardware and also that these softcores are closed source, so modifications or enhancements to these softcores is impossible from a practical standpoint. These constraints prevent the use of FPGAs as prototyping systems previous to the manufacture of an ASIC, or the implementation of one manufacturer's softcore in another manufacturer's FPGA, as for example designers are unable to manufacture an ASIC which includes a Microblaze processor or implement the Nios processor on a Xilinx FPGA. Also, for a designer that tackles low budget projects, the cost of these software packages can be prohibitive.

The solution which is proposed on this article is based on the Leon 2 processor, a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture [5]. The complete sources of this processor are distributed under the GNU LGPL license, together with the needed tools to correctly configure the model.

The synthesis of the Leon 2 processor has been demonstrated on the devices of the most important brands of FPGAs, such as Atmel, Xilinx, Altera and Actel. It also has been optimized for and implemented in several ASIC technologies.

Furthermore, the model also incorporates an implementation of the AMBA (Advanced Microcontroller Bus Architecture) 2.0 bus, which allows for easy addition of custom peripherals to the system [1].

For the software part of the embedded system, we choose to use Linux, as it is an open source system with great reliability and and flexibility, as will be commented further on this article.

For these reasons, we propose as an alternative for design and implementation of embedded systems a solution based on Linux and the Leon 2 processor. Figure 1 shows the typical structure of an application developed on top of this solution. The license costs of both the hardware IP and the operating

Manuscript received March 20, 2006.

H. Guzmán Miranda, J. Tombs and M.A. Aguirre Echánove are with the Department of Electronic Engineering, University of Seville, Spain. E-mail: {hipolito, jon, aguirre}@gte.esi.us.es).

system are nil, and also the cost of the synthesis software is nullified if the FPGA used is supported by a free of charge, stripped-down version of a FPGA design suite, such as ISE Webpack or Altera Quartus II Web Edition.

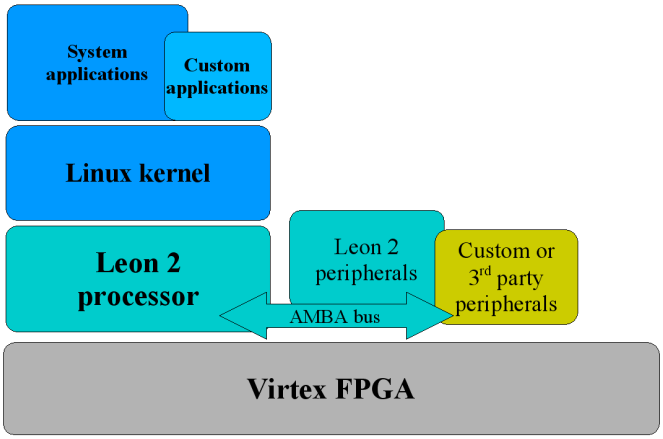

Fig. 1. Block diagram of the proposed solution.

II. Platform

The platform over which the system has been implemented is the XSV-800 development board, manufactured by Xess [6]. The platform makes it possible to demonstrate a minimal system upon a Virtex XCV800 FPGA, which has approximately 800K equivalent gates, including internal RAMs. Both the FPGA and the development board have been discontinued, and the on-board resources are considerably limited for today's standards.

The development board has only 2MB of SRAM memory, which have to be shared between the RAM image from which the system will boot and the available memory for system processes and applications.

It has been necessary to reconfigure the on-board CPLD in order to allow access through the serial port to the processor that has been implemented on the Virtex FPGA. This is mandatory for our implementation, as the operating system which has been implemented on the board maps its console input and output through the board's serial port.

III. Leon2 Processor

The Leon processor [3] is a synthesizable VHDL implementation of a 32-bit processor compatible with the Sparc V8 instruction set, developed for the European Space Agency and at the present time maintained by Gaisler Research.

This processor has the following remarkable features:

- Separate instruction and data caches
- Alu for hardware multiplication and division
- Interface for floating point units and coprocessors
- AMBA 2.0 Bus
- Interrupt controller
- Flexible memory controller, able to work with SDRAM in 32-bit mode and with SRAM and ROM in configurable modes of 8, 16 or 32 bits
- General 16-bit input and output port
- Two UARTs
- Two 24-bit timers
- Debug Supoort Unit (DSU)

Not only is the Leon processor Open Source, but it is also independent of any specific FPGA technologies.

The AMBA bus makes it very easy to extend the functionality of the processor by adding hardware modules. It even allows for reuse of hardware code, whether it be code previously developed by the system designer or third-party IP modules. The intellectual property market places within the reach of any designer a huge library of different peripherals and data processing units [4, 7]. Due to the great acceptance that the AMBA bus has in the industry, it is easy to find hardware modules prepared for their connection to said bus.

IV. Design Flow

When it comes to creating the embedded system based in softcore, a proper design flow must comprise both the use of hardware synthesis tools and software compilation and generation of the RAM image that the system will execute.

The steps followed in our implementation have been:

- Configuration of the processor model, using the tk-based tools (see figure 2) that are distributed with it, in order to select the adequate attributes for the processor to work appropriately in the target board, and also to configure the needed processor peripherals for the specific application, like the cache memories or the available RAM interfaces.
- Synthesis and implementation of the processor model using Xilinx ISE.
- Programming the hardware design on the Virtex FPGA using the tools provided by XESS for the configuration of the board.
- Configuration of the Linux distribution: kernel version, standard C library and specific applications (also with tk-based tools).
- Compilation of the software and generation of the RAM image that will be programmed in the development board.
- Programming the on-board RAM and booting the Linux system.

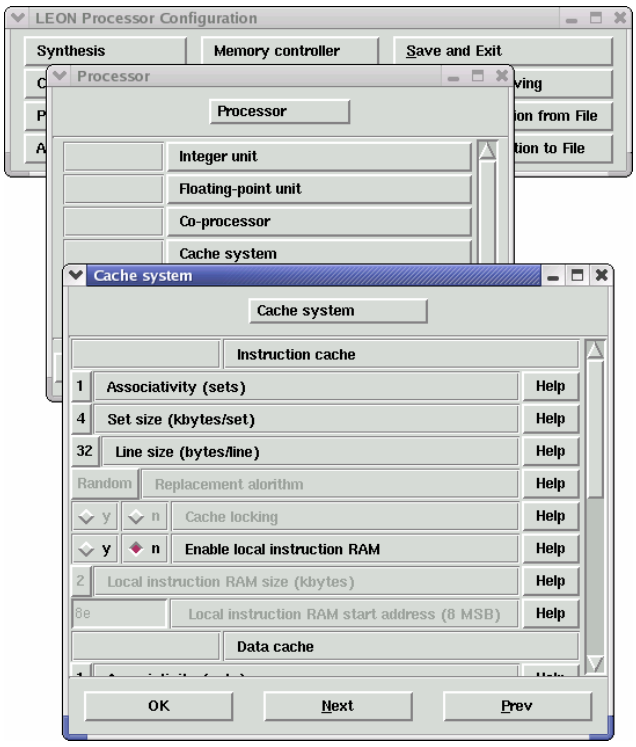

Fig. 2. Configuration tool for the Leon 2 processor. Screenshot of the the configuration of the on-chip cache memories.

V. SNAPGEAR LINUX

Using an operating system in our embedded system, instead of directy programming the needed routines for our application in assembly language, presents a number of advantages, the most important being the ability to have multiple tasks running on the microprocessor.

For our embedded system implementation, the linux distribution used has been Snapgear Linux, a linux distribution specific to embedded systems with support for the Leon processor family.

Linux is a very well known operating system, mature and with support, with multitasking, possibility of real-time operation and many stable applications that can be used without any modifications. Compared to other operating systems for embedded systems, embedded Linux has the advantage of presenting the same interface to the applications than its version for personal computers, so it is possible to use, for the development of the applications, a very similar platform, with the same execution environment and tools that will be available in the embedded system. Development and subsequent debugging of the applications is much easier when the operating system and the execution environment is the same in the development computer and in the final system.

Moreover, Linux, as a UNIX system, is a system so interactive that offers the possibility of carrying out software actualizations from inside the system, that is, if the system has enough memory, it is possible to install the c compiler (gcc) and, with this compiler, rebuild the applications that are to be updated, without having to stop neither the complete system nor the applications that are not modified.

In order for an operating system such as Linux to run properly on a specific microprocessor, usually the processor must have a Memory Management Unit (MMU), which is responsible of giving to each process a memory protected space so each process (including the kernel) is protected against memory corruption caused by other processes. Besides, the MMU deals with the translation of the virtual memory addresses into physical memory addresses before each memory access, presenting to the kernel and processes a standard interface for accessing the memory. Fortunately there are versions of the Linux kernel that have been adapted for their use on processors which do not have an MMU [2]. Snapgear Linux offers the possibility of executing both kernels for systems with MMU and kernels for systems which lack it.

The Leon 2 processor comprises a memory management unit that can be implemented by setting the appropriate option in the processor configuration menu, with a consequent increase of the size of the design. Given that the used platform has some constraints, it has been necessary to synthesize the processor without the memory management unit, therefore the Linux system has been configured to run a kernel with support for non-MMU systems. Specifically, for the implementation described here a kernel 2.0.39 with the aforementioned support has been used.

The configuration tool (see figure 3) allows the designer to adjust the kernel options, as well as the specific applications that will be compiled into the system, so the designer can free some memory space by leaving out the features that are not needed.

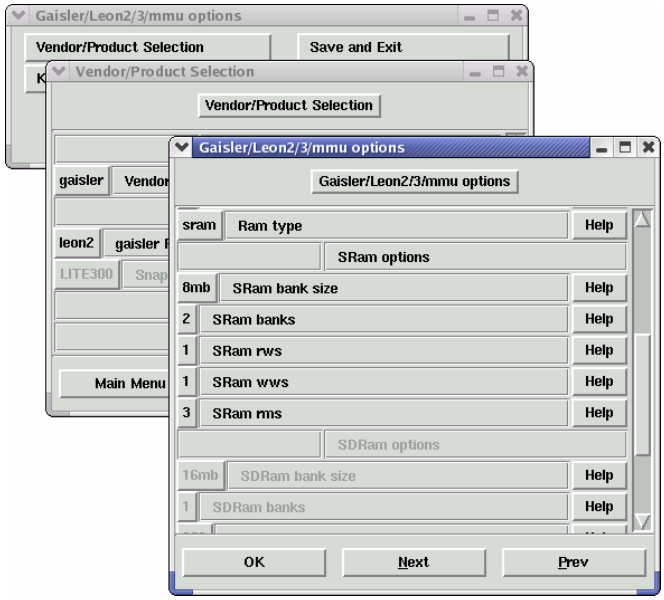

Fig. 3. Configuration tool of Snapgear Linux. The image shows the memory controller configuration menu.

VI. RESULTS

A minimal Linux system has been implemented, the code being executed by the Leon 2 processor synthesized in the FPGA of our development board. Figure 4 shows the console output of the system.

The FPGA occupation, after a low optimizing effort synthesis, is 76% of the slices, which leaves enough resources to add new preipherals for specific applications. The processor runs at a clock speed of 20MHz.

The size of the RAM image is about 600KB, therefore there is more than 1MB of RAM available for system and user applications. This experience has proved that at least 1MB of free RAM is needed to run a minimal Linux system. As more applications are added to the system, this requirement will also grow accordingly.

The drhystone benchmark (version 2.1, C language) has been executed in our system, resulting in a performance of 21190.9 dhrystone/s.

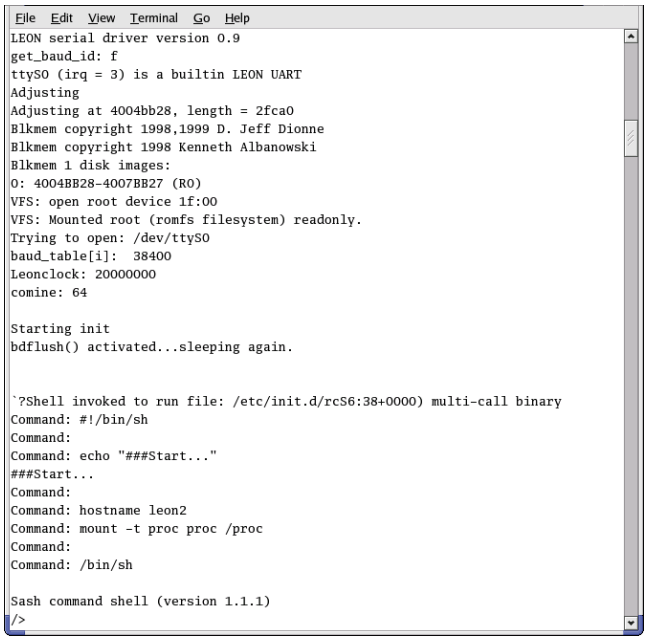

Fig. 4. Screenshot of the minicom program, which is used to establish a serial communication with the embedded system. The figure shows the system shell after a satisfactory boot.

VII. CONCLUSIONS

A minimal embedded system, with appropriate computing power, has been demostrated on a limited hardware platform, without incurring in great license costs. Nowadays the FPGAs supported by the free version of Xilinx Webpack are powerful enough to support the Leon 2 processor. This allows small and medium-sized business and development groups to experiment with this sort of solutions without incurring in any more costs than those of the hardware platform which are, on the other side, unavoidables.

In a platform which had more available memory, it would be possible to easily add dependable support for TCP/IP, therefore an embedded solution based in Linux and the Leon processor could establish network connections, or even be controlled from a web interface for configuration and on-line maintenance. For applications that make use of the TCP/IP protocols, the synthesis of the MMU is advisable, because these applications entail much more system load and, without the MMU, the performance would be heavily degraded.

Furthermore, the possibility of reusing hardware and software code allows for a drastic reduction of development times.

REFERENCES

[1] ARM Limited. AMBA Specification, Rev 2.0. ARM IHI 0011A. May 1999.

[2] Durrant, M. and Leslie, M. How uClinux provides MMU-less processors with an alternative. Embedded.com, December 2002.

[3] Gaisler Resarch. Leon2 Processor User's Manual, Version 1.0.30, XST Edition. July 2005.

[4] Gaisler, J. An open-source VHDL IP library with Plug&Play configuration. IFIP Congress Topical Sessions 2004: Toulouse, France.

[5] SPARC International, Inc. The SPARC Architecture Manual: Version 8 Revision SAV080SI9308.

[6] X Engineering Software Systems Corporation. XSV Board v1.1 Manual. September 2001.

[7] The Opencores Project web site. [Online]. Available: http://www.opencores.org