REPTAR: A UNIVERSAL PLATFORM FOR CODESIGN APPLICATIONS

Alberto Dassatti, Olivier Auberson, Romain Bornet, Etienne Messerli, Jérôme Stadelmann and Yann Thoma

Reconfigurable and Embedded Digital Systems Institute - REDS
HEIG-VD // School of Business and Engineering Vaud; HES-SO // University of Applied Sciences Western Switzerland
Route de Cheseaux 1, 1401, Yverdon-les-Bains, Switzerland
phone: + (41) 24 557 61 60, fax: + (41) 24 557 62 64, email: name.surname@heig-vd.ch
web: www.reds.ch

ABSTRACT

Embedded systems are shaping a new world. There is no sector immune to their adoption and the effects in the long term are unpredictable and fascinating. Embedded systems are designed by embedded system engineers. What technical education and what kind of practical skills these new engineers will need? This is a complex and unanswered question. In this paper we describe our proposal to equip engineering students with knowledge and experience: REPTAR. REPTAR (Reconfigurable Embedded Platform for Training And Research) is a feature rich complex embedded system designed for giving the opportunity to tomorrow's engineers of having hands-on experience on modern technologies and learning by doing. As a side effect REPTAR revealed itself as an invaluable tool for rapid prototyping and research explorations.

1. INTRODUCTION

Back in year 2011, professors at the Reconfigurable and Embedded Digital Systems (REDS) Institute faced critical issues in the labs they were teaching for different courses (digital systems design, embedded programming, ...) as the hardware platforms they were using were either obsolete from a technological point of view or approaching the end of life and no longer available from commercial distributors. To tackle this problem two solutions were envisioned:

1. Replace all development kits used in the different courses with newer ones from multiple providers
2. Design our own home-grown platform which would fulfill the needs of all labs currently given at the institute

After some initial investigations on existing solutions we took up the challenge and the latter solution was accepted. This choice gave us full freedom and flexibility on the design and was also an excellent opportunity for the team of engineers involved in the development process to bring together all the skills they gathered from multiple research projects into a single common realization.

Requirements and specifications of the complete platform were identified and formalized. The platform needed to be very flexible and modular to cope with the variety of topics we are teaching. Hardware platform design, from schematics to production, software development and board bring up took almost a full year. REPTAR was first introduced in labs with students in fall 2012. In this paper we introduce the interested reader to REPTAR, its software and hardware components as well as how we exploited it for research in the last years and how we are going to continue in this direction.

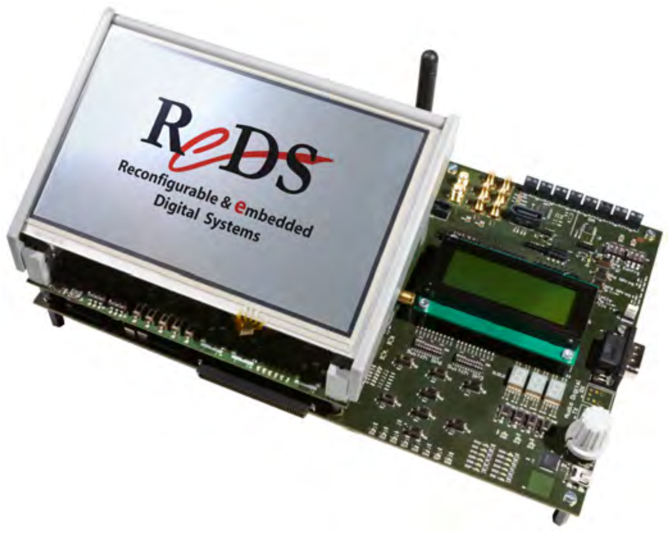

Figure 1: REPTAR System

The rest of the paper is structured as follows. In section 2, the REPTAR embedded platform is introduced, section 2.1 provides more details about its hardware features and the challenges in its development. In section 2.2, the different software environments are described while section 3 reports about some experiences with the board in two main contexts: research 3.1 and education 3.2. Finally, a conclusion paragraph summarizes our findings.

2. EMBEDDED PLATFORM

REPTAR is a universal board for courses in computer architecture and embedded processor-based systems. The full system, visible in Figure 1, combines a Texas Instruments DM3730 SoC [18] (itself consisting of an ARM Cortex-A8 coupled with a TMS320C64x+ DSP) with a Xilinx Spartan 6 FPGA [23]. The platform also includes a large number of control, display and communication peripherals. Modular by design, it offers many possibilities for expansion and usage. The main components can work in standalone mode or coupled, giving the option of developing only on the CPU, only on the FPGA or combine the two. On the CPU side more flexibility is granted by many different development frameworks ranging from bare metal code to single task boot loader, from minimalist RTOS to fully flagged Linux distribution.

REPTAR is not the only system composed by the combination of a processor and an FPGA, there are many similar commercial and accademics systems (an updated list can be found in [13]). The most notable among them is the Zynq Platform [24] from Xilinx. Both systems are composed by a processor and an FPGA tightly coupled, but there are few remarkable differences. First of all when the project REPTAR started, back in 2011, Zynq was only announced[1] and a comparison was not possible; From the architectural point of view there are three major differences between the two platforms:

- Zynq is an ARM dual-core processor equipped with an FPGA, this implies the CPU has to be programmed in order to use the FPGA; this is not the case in REPTAR;
- secondly, REPTAR integrates a powerful programmable DSP core, unavailable on the Zynq (it provides a second ARM core);
- finally, the coupling between the FPGA fabric and the processor, as well as the achivable data rate between the two, is much higher in the Zynq.

These factors made the two systems quite different and very hard to compare. If we consider the REPTAR system globally, the comparison is even less significative.

The following two sections provides greater details about the two sides of the platform, its Hardware and Software components.

2.1 Hardware Architecture

REPTAR platform is composed by two boards: the CPU board (with a Davinci DM3730 processor module [20]) and the FPGA board (equipped with a Spartan 6). The latter is considered as the main board of the entire system and provides all the power supplies. The CPU board is considered as a daughter card. Through this choice, we were able to offer a flexible and evolutive solution with reasonable costs of redesign and update for each part. Figure 2 presents a functional block diagram of the platform.

The processor module features a Cortex-A8 $1GHz$ with $256MB$ of DDR2 running at $800MHz$ and provides a direct support to the Touchscreen ($7''$ capacitive) and the SD-card and USB communication subsystems. Networking interfaces, $100Mbit$ Ethernet link, a Wifi/Bluetooth module and a GPS/GSM card are also directly accessed from the CPU. The FPGA Xilinx Spartan6 XC6SLX150TFGG900-3 offers high-speed Transceivers up to $3GHz$ usable for SATA and other high speed protocols, and more than 500 user programmable IOs (FMC, JTAG, GPIOS). Finally, 4 ADCs and 4 DACs converters give access to the analog world.

One of the key decision in this design was the CPU choice. Many features of the DM3730 have guided our preference for this processor. First, it was completely in accordance with our research and teaching needs in regards of its flexibility and power. We were looking for a processor for a long term solution, with a powerful ARM processor and simultaneously, offering the capabilities of specialized computation as digital signal processing. Second, its internal architecture allowing to interface an FPGA directly on its main bus fulfilled exactly the requirements for the REPTAR platform. Finally, the kind of topics covered by our classes require very detailed knowledge of the CPU architecture: TI provides accessible, updated and complete documentation for this processor.

[1]actual documentation date August 2012.

The PCB (Printed Circuit Board) routing of both boards with so many components was one of our challenge in terms of complexity. Respecting both the electrical (analog and digital) and mechanical constraints, a PCB of 18 layers has been designed for the FPGA board.

The CPU-FPGA communication, a very important part of this mix-platform, can be exploited through two protocols. An asynchronous access allows to easily execute read/writes to/from FPGA registers. A synchronous protocol (the clock being supplied by the CPU) allows to manage burst accesses, the FPGA being seen as a DDR memory. This mode is particularly useful for applications requiring higher throughput than simple command registers accesses, as the throughput can attain a data rate of up to $560Mb/s$ instead of $100Mb/s$ provided by asynchronous transfers. The FPGA reference design already takes care of these protocols, as well as many other low level details such as interrupt generation, and provides a clear and easy to use interface for the integration of a custom design.

2.2 Software Components

A broad portfolio of software components was developed or adapted from existing open source projects. These components can be classified in different categories, namely:

- Adapted development, test and debugging tools;
- Low-level software: boot loader and OS-less devices abstraction layers;
- Operating systems: general purpose and real-time operating systems;
- User space/applications;

To reduce the burden generally tied to the development of embedded systems and make REPTAR an easy and quick to take up platform, a comprehensive framework was set up. First, a version of the open source Qemu emulator [14] was adapted to emulate the core CPU as well as many peripherals of the CPU board. These adaptations resulted in a new 'reptar' machine in Qemu which can easily be started with a `-M reptar` command line option. A basic emulation of the FPGA interface (registers interface) was also added to allow the emulation of the FPGA features. A graphical companion front-end in Qt was developed to allow better emulation of user-facing I/O components. Buttons, switches, LEDs and the touchscreen are currently supported by this extension.

As for many open source embedded systems, U-Boot [6] was chosen as default boot loader. Based on the version available for the CPU module used on REPTAR a customized version was developed. The modifications focused on support of the FPGA through the local bus of the DM3730 as well as on the integration of an exhaustive test framework based on ITBOK [19] allowing early tests of all hardware functionalities. Complementary to the low-level software interfaces provided by U-Boot, a set of standalone toolboxes have been developed to allow the development of bare-metal/OS-less applications. Toolboxes are available for the basic subsystems of the SoC (system initialization, timer, UART, GPIO) and for more advanced blocks such as the LCD screen controller. These components are typically developed and supported under the Texas Instruments Code Composer IDE.

To provide the best suited operating system for any research project or students' lab, several operating systems are supported on the platform. Currently, Linux is supported as

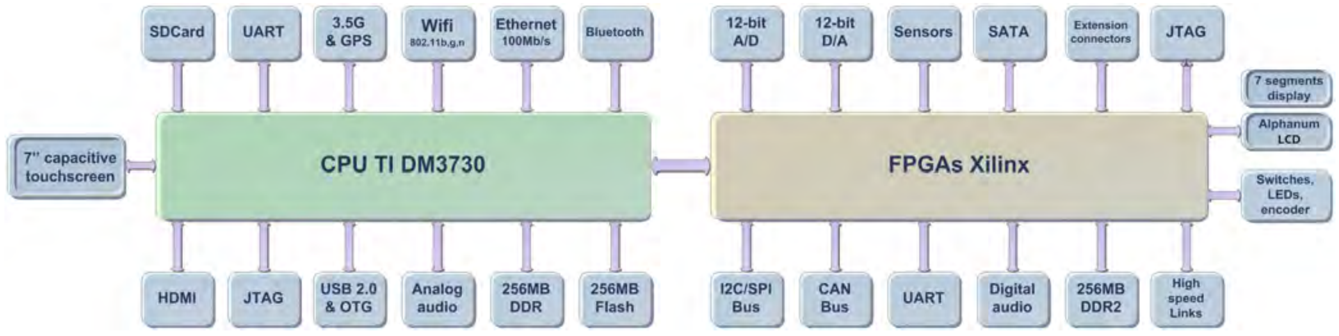

Figure 2: REPTAR block diagram

the reference general purpose operating system. For scenarios where real-time constraints may be required, a Xenomai patched Linux kernel is also available. Further details on this are given in section 3.2 . Work is currently undergoing to further support ChibiOS [2] and RTEMS [12] as real-time operating systems. These operating systems will be used in the real-time programming class with the aim of giving the students a larger view on alternatives usable when the processing power is more constrained.

A Linux kernel is of little use without a consistent and featureful user space. REPTAR supports two kinds of root file systems: Ubuntu core image augmented with cherry-picked packages and a Buildroot [1] based rootfs compiled entirely from sources. The Ubuntu file system offers a standard graphical desktop environment (LXDE) with applications for managing user-friendly interactive usage. Packages installed by default include a graphical network configuration tool, a file explorer and audio/video players. Additional packages can be installed easily using Ubuntu's packages management tool, APT. The Buildroot based rootfs offers more flexibility and level of control than Ubuntu. Being based on known configuration tools (same Kconfig as Linux kernel) and easily tunable, Buildroot revealed itself as the ideal build system for our modular platform. A standard Buildroot configuration is available from our build framework. Students and researchers can easily take over this basic configuration and tune it by adding applications or libraries specific to their project. To further broadens the portfolio of available user space stacks, ports of Android and Tizen [8] are currently ongoing and will soon be available.

3. APPLICATIONS

As for our initial goal, REPTAR filled the need of a rapid prototyping platform for our research projects as well as a feature rich platform for education. In the following we will report about some projects and courses examples where this platform proves its value.

3.1 Research

Our research focus is on high performance embedded systems. In this context the availability of a fast prototyping platform, rich of interfaces and with a rich set of software components is definitely a valuable asset. The next sections will provide an overview of some of our researches demonstrated on REPTAR.

3.1.1 SOSoC

Nowadays, technology advances in the design of general purpose System on Chip have revolutionized the conception of embedded systems. Beside of the General Purpose Processor, we can often find other dedicated processors and co-processors like, in the case of the REPTAR platform, a DSP and a NEON Single Instruction Multiple Data co-processor. The SOSoC project allows the application developer to use those different calculation capabilities in a simple and transparent way, even with minimal knowledge about the hardware.

For this project, a framework supplying optimized functions for specific targets has been developed. It does not only provides specific algorithms, but also adds an abstraction layer between the user application and the hardware specific functions. In the facts, the application is only coded and compiled once, without even worrying about the hardware modules available. The SOSoC framework allows then to dynamically execute its functions on the most adapted processing unit. This could be achieved by the library dynamically monitoring all the different function calls with their associated parameters; consulting this information the system can select the most promising solution in every situation and adapt the dispatching strategy to variable loads. This framework has been thought to be easily expandable, by adding new optimized functions, or new processing units, or even to be ported on another hardware. Figure 3 shows the architecture of SOSoC. It is divided into several libraries, one generic containing the general dispatching framework, and a specific library for each computation unit supported. We can also see that the communication with the DSP is done using the SysLink [17] capabilities. For the interested reader, more information on the SOSoC project can be found in [11].

In this context, REPTAR played the real-time demonstrator of our approach. The demo software applies different filters on a video stream and then displays the processed output on the embedded screen. At the launch of the application, all calculations are made on the ARM processor alone, without any other optimization than the compiler's ones. Then, the performances can easily be enhanced by joining the NEON co-processor capabilities to the GPP or even by adding the DSP. All this can be done at run-time, automatically or instructed by the user.

3.1.2 EmbeddedXen

Virtualization is a technique that has been used for many years in the field of large IT systems and high-reliability

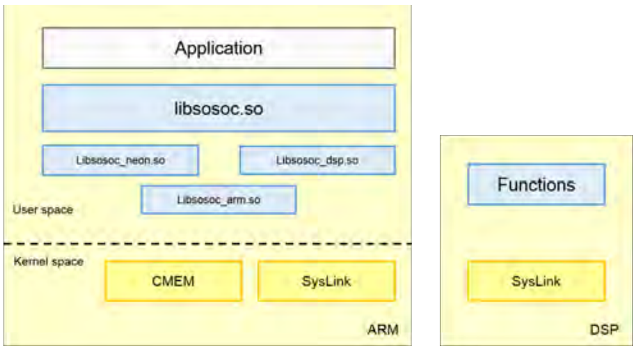

Figure 3: SOSoC architecture diagram

servers. By isolating run-time environments from each others and from hardware resources, virtualized environments provide a high level of security, reliability and flexibility. EmbeddedXen, a virtualization framework for embedded systems based on the well-known Xen project [9], is the result of several years of research in the field of virtualization tailored to embedded systems. Being developed for ARM architectures only, EmbeddedXen was a framework of choice to port on REPTAR. EmbeddedXen is currently supported on REPTAR with two guest domains: A privileged domain (dom0) and a non-privileged domain (domU) share the hardware resources exposed by the platform. This flexible infrastructure allows exploration and research in the fields of security, heterogeneous platforms support and innovative hardware-software interactions. Projects using EmbeddedXen are currently ongoing to develop new architectures for smart energy management and secure mobile environments. The interested reader will find more details on EmbeddedXen architecture in [4] and at EmbeddedXen project's page on SourceForge [5].

3.2 Education

REPTAR proved to be a versatile and practical instrument in many courses. The topics that can be experimented on the board range from DSP programming to FPGA design, from HW-SW partitioning to Operating Systems, from Multi-Processor systems to real-time signal processing. In the three following examples we describe how we used REPTAR to improve our educational offer. They are a small subset of the classes using REPTAR. For a more complete and up to date list please refers to [15]. On the same site all contacts details are given if you need more information or you are interested in using REPTAR in your class or project.

3.2.1 Real-time programming

Real-time programming is an extremely challenging field in computer science. Students have often difficulties to figure out how the theoretical concept maps to physical systems; Complexity of modern microprocessor based systems increase the complexity of the topic and makes practical experience quite limited. The introduction of laboratories based on REPTAR redesigned the scenario. REPTAR and its combination of microprocessor and FPGA gives the opportunity of comparing the different granularities in time mastering as well as the possibility of interfacing easily with the real world. In order to set up this labs we provided some basic setup for the students to experiment with. The first one is based on standard Linux kernel, the second on the RT patch set [3] and the third on Xenomai [22]. Following a set of exercises, students walk through the details of the different implementations and measure the real-time performance acquiring hands-on experience. In the last part of the class the students are often confronted with a complex real-time problem where they are asked to perform a multi task elaboration of signals coming from external components connected to the board (usually using the ADC and DAC accessed through a RTDM driver [21]). REPTAR is in this case a major motivation factor and stimulus to acquire in-depth knowledge of the topic.

3.2.2 REPBOT

REPBOT - contraction of REPTAR and robot - is a robotic platform developed by students in the frame of the Embedded Systems Design course, a third year course for students in Computer Engineering. Based on a REPTAR board and a commercial robot's chassis, students developed a complete robotic platform. In teams of four or five students the goal was to design and implement an embedded system capable of driving an autonomous robot. Apart from the base building blocks, no constraint was specified and students could organize their work as they wanted to reach the final goal. From mechanical engineering to hardware extensions design or low-level programming for the CPU or the FPGA to control actuators and sensors, the team splitted the work in different work packages, each student being responsible of one or several of them. The concrete application of concepts and techniques learned in various courses was a real motivation for the students who developed two different functional platforms. With these highly valuable results, they presented their robot at the Swiss Informatics Competition 2013 and were awarded the third place for their realization [16].

3.2.3 Mandelbrot set calculator

The REPTAR platform can be exploited for co-design projects, notably to develop hardware accelerators. In that context, a Mandelbrot set calculator (written in VHDL) has been developed. A Mandelbrot set [10] is a 2-D drawing representing the complex plane on which each pixel corresponds to a complex number of the plane. For each pixel, the function $Z_{n+1} = Z_n^2 + C$ is recursively applied, C being the pixel coordinate, Z_0 being 0. If the series diverges, the pixel is considered to be outside of the Mandelbrot set, else it is inside. This process requires to fix the maximum number of iterations to be performed to approximate the real set, and obviously, parallel processing can be applied, as every pixel is independent of the others.

A C++/Qt application has been developed and allows to display the Mandelbrot set on the REPTAR touchscreen, exploiting floating points or fixed number arithmetic (32, 16, or 8 bits). From this, an FPGA calculator has been designed. The software only sends to the FPGA the information concerning the bottom-left point, the number of pixels on the two axes and the pixel size. The FPGA design then has to perform the calculation and returns, for each pixel, the number of iterations reached as well as the last Z_n. These data are then used to display the graphical Mandelbrot set on the REPTAR screen. This co-design application is interesting in the sense that it shows a limited data transfer from the CPU to the FPGA (5 words), and a big one in the opposite direction ($3 \times S_x \times S_y$ words, where S_x and S_y represent the size of the image) . The software code can be totally agnostic relative to the FPGA design implementation, that can be purely sequential or totally pipelined. Therefore it allows for any kind of implementation on the hardware part. It would also be possible to let students rewrite the software drivers responsible for accessing the Mandelbrot accelerator.

Finally, we can notice that this application can easily be adapted to any "iteration based" mathematical curves like Julia Sets, or the Newton fractal [7]. This application, being the first hardware accelerator will be followed by others, letting students embed code on a real FPGA with display possibilities offered by the REPTAR touchscreen.

4. CONCLUSION

Embedded systems teaching, being on the software or hardware side, requires the students to put their hands on real hardware, not only simulators. In that context, REPTAR is the perfect platform for a rich set of courses, allowing the students to play with the same hardware through many courses. Its usefulness is already demonstrated, and new labs are developed every semester. Its complexity is such that many peripherals are embedded on board, and that it avoids the use of different boards for each course.

On the research side, this platform already showed an excellent potential, as explained in this paper. The full understanding of the hardware and the software (both low and high level) allows the REDS institute to easily start new projects on REPTAR, or to use it as a demonstrator for specific applications.

Finally, the REDS institute is definitely open to collaborations regarding the development of the board itself or the applications. The Gerber files can be asked so that another institution could directly build new boards without further development, or collaborations can be envisioned to enhance the board.

5. ACKNOWLEDGMENTS

The authors would like to thank the HEIG-VD for the financial support that allowed the realization of the REPTAR board and all its surrounding development. Their gratitude also goes to all those who participated directly or indirectly to the realization of this platform.

REFERENCES

[1] Buildroot. Buildroot making embedded linux easy. http://buildroot.uclibc.org/.

[2] ChibiOS. Chibios/rt. http://www.chibios.org/dokuwiki/doku.php.

[3] CONFIG_PREEMPT_RT community. Real-time linux wiki. https://rt.wiki.kernel.org/index.php/Main_Page.

[4] D. Rossier. EmbeddedXEN: A revisited architecture of the XEN hypervisor to support ARMbased embedded virtualization. Technical report. http://upload.wikimedia.org/wikipedia/commons/5/58/EmbeddedXEN_publication_final.pdf.

[5] D. Rossier. EmbeddedXEN virtualization framework. http://sourceforge.net/projects/embeddedxen/.

[6] denx. Das u-boot – the universal boot loader. http://www.denx.de/wiki/U-Boot.

[7] B. Epureanu and H. Greenside. Fractal basins of attraction associated with a damped newton's method. *SIAM Review*, 40(1):102–109, 1998.

[8] Linux Foundation . The os of everything. https://www.tizen.org/.

[9] Linux Foundation. The xen project, the powerful open source industry standard for virtualization. http://www.xenproject.org/.

[10] B. Mandelbrot. *Fractals: form, change and dimension*. W.H. Freeman, 1977.

[11] O. Nasrallah, W. Luithardt, D. Rossier, A. Dassatti, J. Stadelmann, X. Blanc, N. Pazos, F. Sauser, and S. Monnerat. Sosoc, a linux framework for system optimization using system on chip. In *SOC Conference (SOCC), 2013 IEEE 26th International*, pages 284,289. IEEE, 2013.

[12] OAR Corporation. Real-time executive for multiprocessor systems. http://www.rtems.org/.

[13] Philip Freidin. Fpga boards and systems. http://www.fpga-faq.com/FPGA_Boards.shtml.

[14] QEMU. Qemu. http://wiki.qemu.org/Main_Page.

[15] REDS. Formations. http://reds.heig-vd.ch/formations.

[16] SI. Wettbewerb sic (swiss informatics competition) 2013. http://www.s-i.ch/veranstaltungen/wettbewerb-sic-2013/.

[17] Texas Instruments. http://processors.wiki.ti.com/index.php/Category:SysLink.

[18] Texas Instruments. Dm3730 digital media processors (rev. d). Technical report. http://www.ti.com/product/dm3730.

[19] Texas Instruments. Itbok. http://processors.wiki.ti.com/index.php/ITBOK.

[20] Variscite. Var-som-om37 cpu : Ti am3703 / dm3730. http://www.variscite.com/products/system-on-module-som/cortex-a8/var-som-om37-cpu-ti-am3703-dm3730.

[21] Xenomai. Real-time driver model. https://rt.wiki.kernel.org/index.php/Main_Page.

[22] Xenomai. Xenomai: Real-time framework for linux. http://www.xenomai.org/.

[23] Xilinx. Spartan-6 fpga product brief. Technical report. http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/.

[24] Xilinx. Zynq-7000 all programmable soc. http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/.