TULIPP

H2020-ICT-O4-2015

Grant Agreement n° 688403

D3.1:
Low power HAL board support development report

Lead Author: Antonio Paolillo, HIPPEROS S.A.

with contributions from:

Organisation no.	Organisation name	Participant Name
4	HIPPEROS S.A.	Antonio Paolillo
4	HIPPEROS S.A.	Christian Lemer
4	HIPPEROS S.A.	Ben Rodriguez

Reviewers

Organisation no.	Organisation name	Participant Name
5	FRAUNHOFER	Boitumelo Ruf
5	FRAUNHOFER	Tobias Schuchert

Document description

Deliverable number	D3.1
Deliverable title	Low power HAL board support development report
Work Package	WP3
Deliverable nature	Report
Dissemination level	Public
Contractual delivery date	January 2017 (M12)
Actual delivery	January 2017 (M12)
Version	1.0

	Written by	**Approved by**
Name Signature	Antonio Paolillo	

Version history

Version	Date	Editors	Description
0.1	2016/11/24	Antonio Paolillo	First draft
0.2	2016/12/20	Antonio Paolillo, Ben Rodriguez, Christian Lemer	HIPPEROS Internal review
0.3	2017/01/09	Antonio Paolillo	First modifications according to review from SUNDANCE
0.4	2017/01/10	Antonio Paolillo, Boitumelo Ruf, Tobias Schuchert	Integration of FRAUNHOFER review
1.0	2017/01/19	Antonio Paolillo	Document finalisation

Abstract

In this deliverable, we present the results achieved in task T3.1 "Low power HAL board support development". In the context of the Tulipp Reference Platform instance, the main goal of this task is the integration of the board supplied by SUNDANCE as a target platform supported by the HIPPEROS RTOS. SUNDANCE supplied the EMC2 board, a mother-board containing a "Trenz Electronic TE0715" daughter-board. This board integrates a Zynq-7000 System-on-Chip (SoC). The Zynq-7000 is a heterogeneous chip that embeds both a multi-core ARMv7 Cortex-A9 CPU (to run software tasks) and a FPGA part (to run hardware accelerators). The main work achieved is the support of this processor platform for both mono- and multi-core versions of the HIPPEROS kernel for all supported memory models. We considered only the Processing System of the Zynq SoC in this task. The Zynq Programmable Logic will be considered in future tasks (T3.2).

Table of contents

List of figures

List of abbreviations

BLING	Bare metaL unIt testiNG
BSP	Board Support Package
FPGA	Field-Programmable Gate Array
FSBL	First Stage Boot Loader
HAL	Hardware Abstraction Layer
HIPPEROS	High Performance Parallel Embedded Real‑time Operating Systems
ISA	Instruction Set Architecture
MMU	Memory Management Unit
MPCore	Multi-Processor Core
OCD	On-Chip Debugging
PL	Programmable Logic
PS	Processing System
RTOS	Real-Time Operating System
SoC	System-on-Chip
UART	Universal Asynchronous Receiver/Transmitter

1. Introduction

The goal of Work Package 3 (WP3) are the following [1]:

- to design and develop a parallel real-time operating system that can handle the target platform specifics and constraints and able to optimize low-power processing;
- to create the required standard APIs and runtime libraries for the RTOS;
- extend the operating system with support for the hardware and software requirements of image processing;
- instantiate the operating system on the Tulipp Reference Platform.

The deliverable D3.1 presents the results achieved within task T3.1 "Low power HAL board support development". The goal of this task is to implement and instantiate the low-level parts of the reference RTOS. This is required as the RTOS has to be integrated with the rest of the components of the Tulipp Reference Platform, especially the hardware target. Therefore, this task defines hardware-to-OS interfaces to be able to build, deploy and debug RTOS with applications on the hardware selected for the Tulipp platform.

The instantiation efforts were mainly focused on the support of the Xilinx Zynq-7000 processor as a target platform for the HIPPEROS RTOS. This support will enable the first instantiation of the Tulipp Reference Platform. This is the SoC embedded in the development board provided by Sundance to be an instance of the hardware component of the Tulipp platform.

Results achieved consist in the support of the complete RTOS, i.e. a multi-core version supporting all memory models: no MMU, single page table and full process isolation.

As most of the design of the HIPPEROS kernel is hardware-agnostic, only the parts that truly depend on the selected platform had to be extended. Other parts (the higher layers of the kernel and user space code) have been imported from the existing HIPPEROS code base. This design ensures flexibility of the target OS as required by the interfaces of the Tulipp Reference Platform.

The extended parts, as shown in Figure 1, are:

1. The Hardware Abstraction Layer (HAL) implementation: one of the modules of the kernel, the HAL is the module closest to the hardware as it implements all the hardware primitives required by the rest of the kernel. To implement the HAL the source tree is divided in two categories:
    a. Architecture related code which provides HAL primitive implementations through features provided by the Instruction Set Architecture (ISA) or the processor family. In this case, the processor is a Cortex-A9 MPCore (with two cores) that implements the ARMv7-A ISA.
    b. Processor related code which provides HAL primitive implementations through features provided by the System-on-Chip. In this case the processor is based on the Xilinx Zynq-7000 family.

2. The Board Support Package (BSP) is a package containing all the tools, scripts and binaries required to build, deploy and run the HIPPEROS RTOS image on the target hardware. It also includes tools to debug both the HIPPEROS RTOS and the supported applications remotely on the embedded target. It is part of the Tulipp Reference Platform and consists of several hardware-to-OS interfaces.

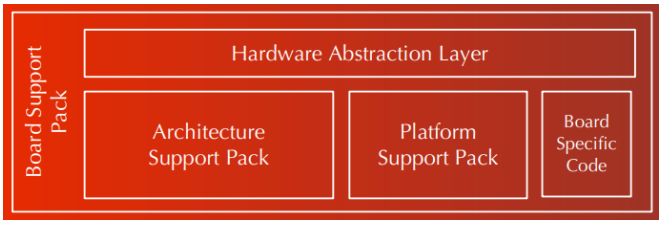

Figure 1: The HAL and the BSP components.

Part of the work also consisted in improving the design of the Hardware Abstraction Layer interface to ensure flexibility, efficiency and genericity of the system. Indeed, the task T3.1 allowed us to define clearly the limits between the different components of a target support: architecture, platform/processor and board specifics. We also extracted the abstract interfaces existing between hardware and operating systems (OS), that can be applied to any compliant RTOS or target hardware in order to support the Tulipp Reference Platform. Finally, this allowed HIPPEROS to be part of the Tulipp eco-system by providing a RTOS solution compliant to the reference RTOS of the Tulipp Reference Platform.

The next sections present how we developed, tested and validated these components for task T3.1.

2. Architecture support

This section presents the integration of the support of both the ARMv7-A architecture [2] and the Cortex-A9 MPCore processor family [3] which is the architecture of the Processing System of the Zynq-7000 SoC. Figure 2 shows the schema of this kind of processor.

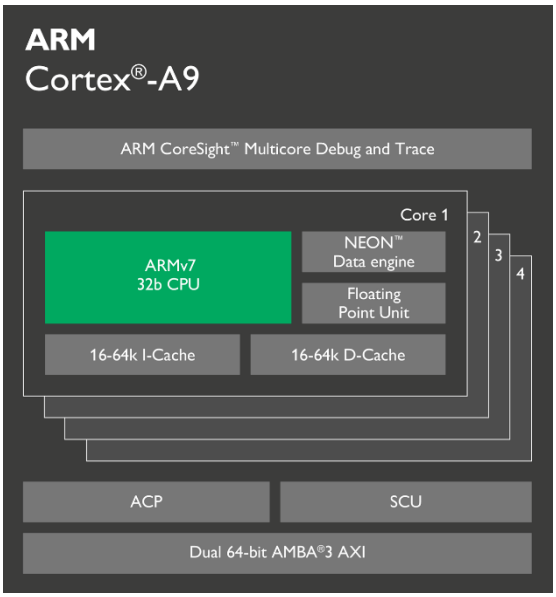

Figure 2: A multi-core Cortex-A9 processor, supporting the ARMv7-A 32 bits architecture.

The HAL primitive components that we had to re-implement at the architecture level are the following:

1. **Start-up code** which consists in low-level (assembly) code to set up the environment to enable the boot procedure of the HIPPEROS kernel.
2. **Interrupt controller and service routines** which allows to configure the machine interrupts with interrupt service routines. These routines are the entry points of the kernel from user space.
3. **Virtual memory management** which allow the configuration of the Memory Management Unit (MMU) and the associated page tables. This enables advanced memory management techniques such as providing private virtual address spaces to user processes.
4. **Cache management** which configures the private caches of the cores. These components allow to clean and invalidate the caches when the kernel requires it.
5. **Timer device drivers** which enable the usage of timers provided by the architecture (in this case, the ARM *Global timer*) allowing to keep track of time and manage events in the operating system.
6. **Atomic primitives** which make use of specific instructions provided by the architecture to perform atomic operations in order to implement critical sections (with semaphores and mutexes).

7. **Floating point support** to take advantage of the Floating Point Unit (FPU) accelerator component. This enables support of user space code that uses floating point arithmetic provided in the instruction set.

Additionally, the build system of the HIPPEROS RTOS requires the usage of a *cross toolchain* (including compiler, linker, binary utilities, etc.) for the C language that support code generation for the target architecture and processor family. We now support the two following toolchains:

1. The **GNU** toolchain that includes the GCC compiler.
2. The **LLVM** toolchain that includes the Clang compiler.

Both toolchains are fully supported for builds of the kernel targeting the ARMv7-A architecture for the Cortex-A9 MPCore processor. They generate binary code for this architecture. The toolchains also support code written in assembly which is present in some parts of the code base (specifically target-specific code). This build process is illustrated by Figure 3. Cross toolchains are part of the hardware-to-OS interfaces defined to ease component integration in the Tulipp Reference Platform.

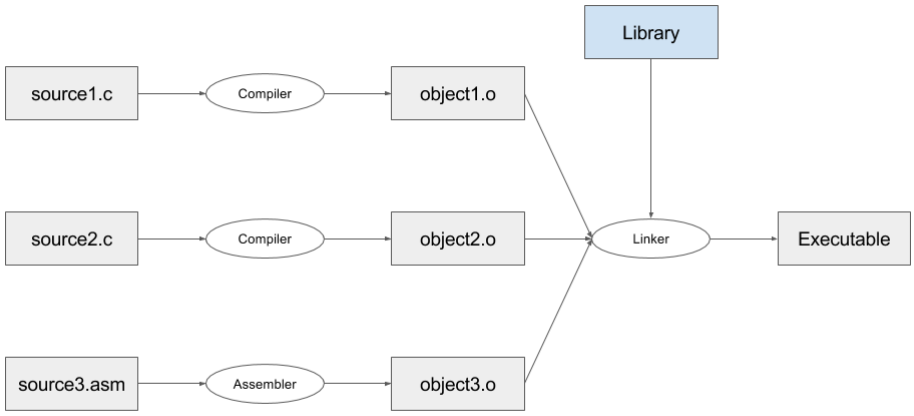

Figure 3: Build system to generate an executable from various source modules. In the case of a cross toolchain, the executable is formatted to run on the remote target.

3. Processor platform support

The board provided by *Sundance* is the EMC²-Z7015 (cf. Figure 4) [4]. It is a mother-board carrying a *Trenz Electronic* daughter-board called TE0715 (cf Figure 5). This board embeds a Xilinx Zynq-7015 processor that belongs to the Xilinx Zynq-7000 processor family [6]. The processor schema is represented by Figure 6.

Figure 4: The Sundance EMC²-Z7015 board.

Figure 5 : The Trenz Electronic TE0715 board.

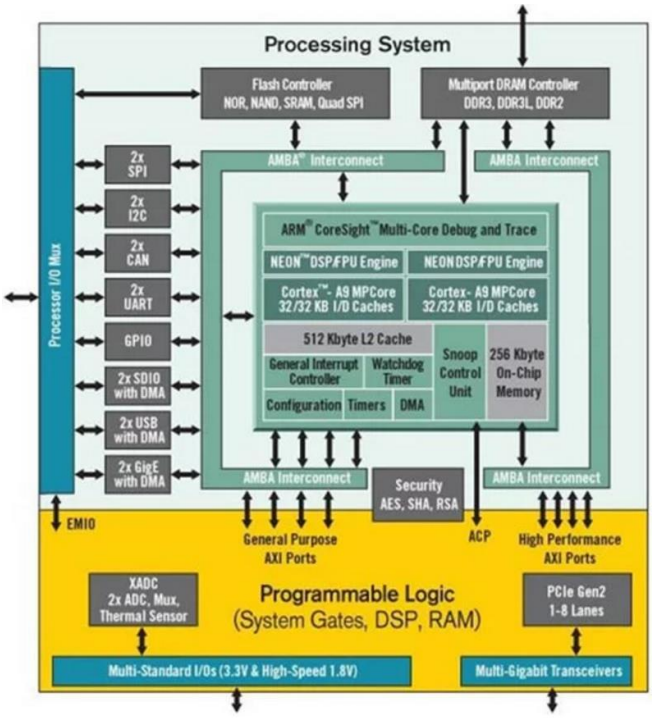

Figure 6 : The Xilinx Zynq-7000 All Programmable SoC schema.

Zynq-7000 processors are heterogeneous platforms containing both FPGA area (called the "Programmable Logic" (PL)) and a multi-core CPU (called the "Processing System" (PS)). For the PS the documentation of the Zynq-7000 provides sufficient information to implement the processor support for the HIPPEROS RTOS. The PS is based on a dual-core ARMv7-A Cortex-A9 MPCore processor (as discussed in Section 2). In this task (and in this document), only the PS is considered. The PL configuration and support by the reference RTOS is part of task T3.2.

In addition to the architecture-provided features, processor component support includes the following features:

1. Implementation of a **logging device**, the UART driver. This allows the RTOS to output logging information through the serial port.
2. Configuration of the **device tree**, which contains all information to configure the implementation of the HAL (random access memory mapping, device addresses, etc.).
3. Secondary cores start-up and inter-processor interrupt management that allows to support the **multi-core** version of the HIPPEROS kernel (enabling parallelism at software level).

To allow more flexibility in the development phase we also implemented support for the emulation of the Zynq-7000 processor on the QEMU emulator [7]. QEMU allows to emulate a wide variety of platforms and we integrated it as a target of the HIPPEROS RTOS. Currently the Zynq-7000 support on QEMU only supports mono-core platforms. Therefore only the mono-core HIPPEROS kernel is available on QEMU. Both mono-core and multi-core builds are available on the physical platform.

We implemented all HIPPEROS virtual memory models on the target:

1. No MMU model: this model provides no isolation and no specific protection (e.g. against kernel over-writing). This is the simplest and most efficient model but also the most unsafe one. However when supporting a new platform/architecture we usually start by this one as it is the easiest to implement. It is illustrated by Figure 7.
2. MMU enabled with a single page table: this model creates a single page table for the whole system. The kernel is protected from user read/write accesses but all tasks share a single address space. User processes and tasks are therefore not protected from each other, while kernel memory is fully protected. It is illustrated by Figure 8.
3. MMU enabled with one page table per process: this model provides full memory isolation and protection. The kernel memory is not accessible from user processes and each process has its own address space. Therefore processes are isolated from each other and cannot access the same regions of physical memory. Communication between processes requires the usage of specific channels provided by HIPPEROS Inter-Process Communication modules. This is the most reliable memory model but requires a large amount of memory to store the page tables for each process. Threads issued by the same process share the same address space (by design). It is illustrated by Figure 9.

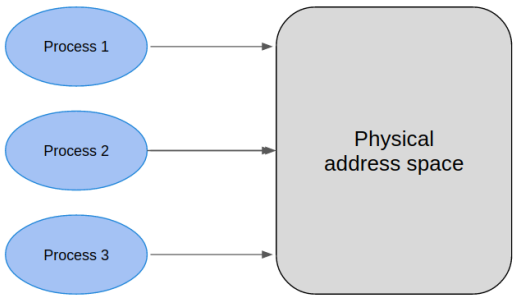

Figure 7 : Memory model when no MMU enabled. Processes directly access physical memory.

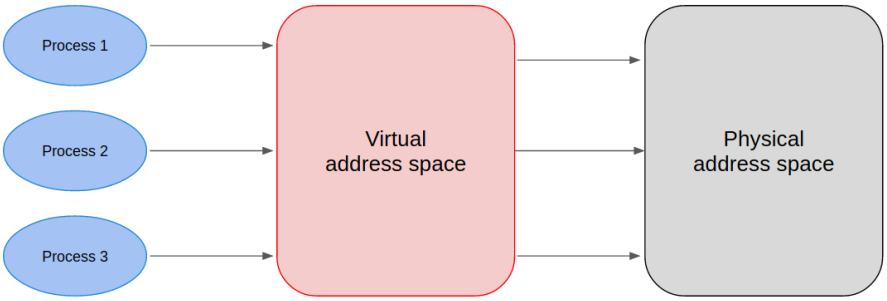

Figure 8 : Memory model with a single page table. It provides memory protection only and processes share the same address space.

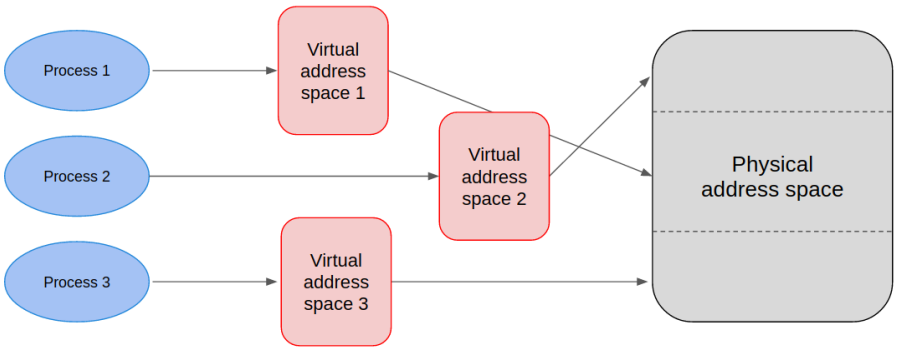

Figure 9 : Memory model with full virtual memory. It provides memory relocation and isolation between processes.

4. Board support

The architecture and processor support sub-tasks aimed at generating the build of the HIPPEROS kernel (and the associated binary image). However in order to deploy, run and debug this image on the target other tools and programs are required. The Board Support Package (BSP) is the set of such tools. This section describes a list of the tools required to implement the BSP. These tools are part of the hardware-to-OS interfaces of the Tulipp Reference Platform. Indeed, both the target hardware and the reference OS have to support a bootloader and a remote debugging/image deployment solution.

First of all, the target requires a boot-loading procedure. This feature is provided by a software called bootloader. The HIPPEROS RTOS supports the U-Boot bootloader, meaning that the HIPPEROS image will interface with U-Boot. Moreover, Zynq processors have specific requirements to boot and so part of the bootloader is provided by Xilinx in the Vivado toolchain [8]. Therefore, the boot-loading procedure of the board has two stages:

1. First Stage Boot Loader (FSBL), very low-level code to initialise the PS and the PL properly.
2. U-Boot, to load and run the RTOS image.

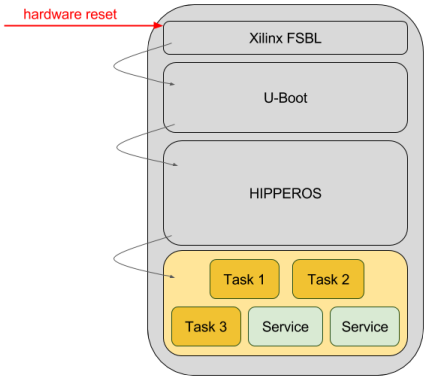

Figure 10 : The boot procedure of HIPPEROS.

The FSBL boots the board and U-Boot which in turn loads the HIPPEROS image on it and jumps into operating system code as illustrated on Figure 10. Therefore, the BSP development included the creation of a U-Boot build compatible with the target. We adapted the configuration of the U-Boot build from the one used for the ZedBoard [9] (which is another development board with the Zynq-7000 platform). We also generated the FSBL image and some headers required by the U-Boot compilation from the Vivado toolchain and adapted them for the Trenz TE0715 board. Both FSBL and U-Boot are written on a SD card that runs the boot procedure at reset time when inserted in the SD port of the board. The whole process to build, configure and install the files on a SD card is automated in a *bash* script (the command language interpreter) provided to the user in the HIPPEROS package.

We also need tools to communicate between the personal computer and the embedded board. This allows to deploy images and to remotely debug code running on the target. We use *OpenOCD* [10], a software allowing on-chip debugging (remotely, from a computer host). OpenOCD requires a physical

connection between the PC and the board through USB/JTAG. Again, we adapted the configuration files for the ZedBoard to create the OpenOCD configuration files for the TE0715 board. It allows to load the RTOS and application images without loading it on a SD card (this enables faster development cycles) and easy on-chip debugging through a remote GDB session. The flow is the following: GDB is running locally on the host computer and connects to OpenOCD which communicates with the board through JTAG. Each GDB command is sent to OpenOCD through TCP/IP (OpenOCD implements a GDB server) and it translates them into JTAG commands. The OpenOCD configuration files we integrated allow multi-core debugging on the PS. We also implemented a software reset command to reset the physical board through OpenOCD. This flow is illustrated by Figure 11.

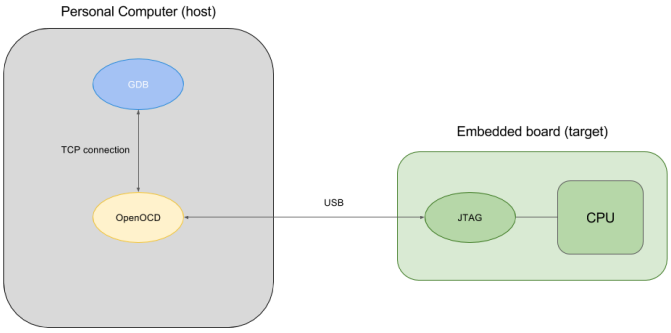

Figure 11 :. Debug flow through OpenOCD and JTAG.

We integrated the build of these new targets in the HIPPEROS build system, which is based on CMake (an open-source cross-platform family of tools to build, test and package software).

5. Hardware

We provide a shipped package containing the RTOS image and the HIPPEROS Software Development Toolkit (SDK). The SDK defines a development flow for a real-time application on the HIPPEROS RTOS. This development flow requires hardware tools that differ for each chosen target platform. This section presents such hardware for the selected hardware target.

In the case of the EMC² board, the hardware requirements are the following:

1. the EMC² board which contains the Trenz TE0715 daughter-board (provided by Sundance);
2. a power cable with a switch (provided by Sundance);
3. a classic power supply for computers;
4. a SD card to store the boot-loaders;
5. a micro-USB cable to output data from the serial port to the computer host;
6. an Olimex JTAG debugger for ARM (https://www.olimex.com/Products/ARM/JTAG/ARM-USB-OCD-H/), that enables OpenOCD communications between the PC (through USB) and the board (through the Xilinx adapter);
7. an adapter connector for Xilinx (http://shop.segger.com/J_Link_Xilinx_Adapter_8_06_19_p/8.06.19.htm)

The whole set-up looks like the picture represented in Figure 12.

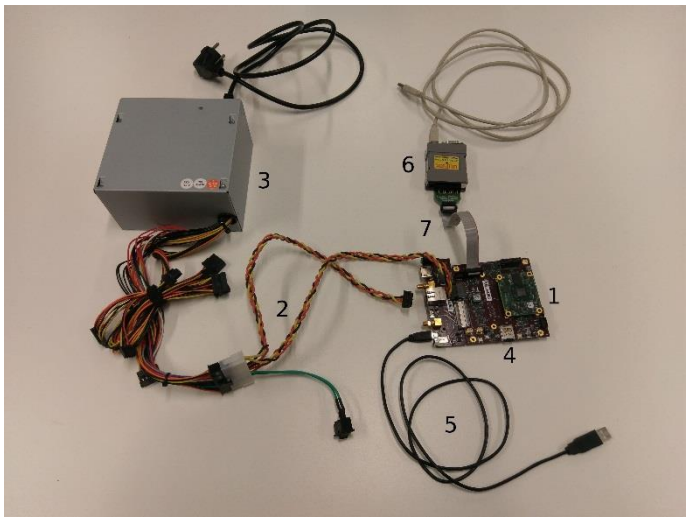

Figure 12 : The hardware set-up of the EMC² board enabled for software debugging.

6. Development lifecycle and product validation

To develop the requested features of task T3.1, we applied standard software engineer methodology. The software lifecycle of HIPPEROS include requirement analysis, design, development and testing to validate the product. This section presents this software lifecycle with an emphasis on the testing methodology. We apply a strict test automation philosophy to avoid regression in our code base and cover the large variety of builds of the HIPPEROS kernel.

In order to validate the integration of the Zynq-7000 processor platform and the TE0715 board as a target of the HIPPEROS RTOS, we added it in our testing system. This section describes the work achieved with regard to the testing and validation of HIPPEROS on the target hardware.

The HIPPEROS operating system software is tested in three ways:

1. Unit testing: unit tests are a set of C++ programs executed on a Linux system (with the Google test unit framework for C/C++). They cover the kernel sections of the code that are hardware-agnostic. Modules tested include process and thread management, data structures, memory management, events management, etc.
2. Platform testing: platform tests are similar to unit tests but they are executed on the target platform. We developed an internal framework called *Bare metaL unIt testiNG* (BLING) that allows to run unit tests on a target platform and retrieve execution reports and results through the serial port in ASCII format. The platform tests are mainly used to validate the HAL introduced in Section 1 (which is not covered by unit tests). This means that platform tests are built for each supported target. They cover mono-/multi-core features, MMU enabling, interrupts, etc.
3. Integration testing: integration tests cover high-level features of the kernel such as system calls and interaction between tasks, processes and threads. They are essentially a build of the RTOS combined with an application (which is composed of a set of software tasks). Builds cover several RTOS versions such as mono- and multi-core and the different virtual memory models introduced above.

All the test layers are executed on a continuous integration test server running Jenkins.

We added platform and integration tests corresponding to the different builds of the RTOS on the TE0715 board and on the QEMU Zynq-7000 virtual target.

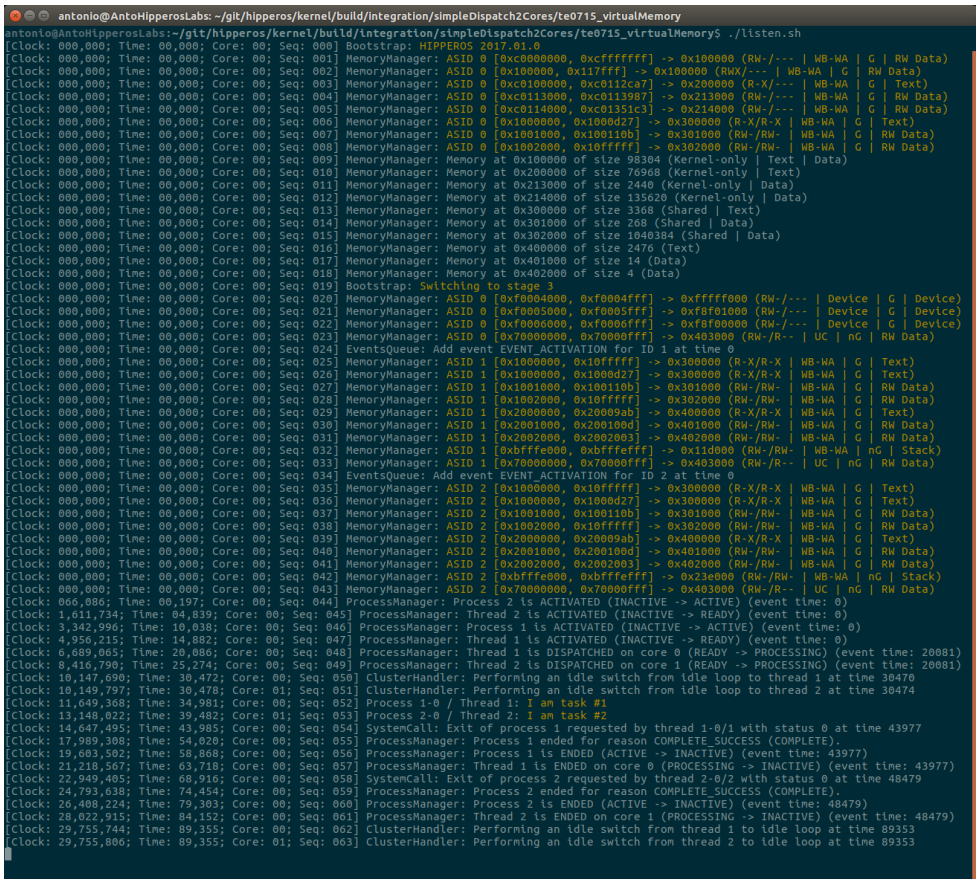

Figure 13 : Console showing the execution of a multi-core integration test

7. Conclusions

This deliverable D3.1 described how we achieved the goals of tasks T3.1 called "Low power HAL board support development".

We implemented a flexible and reliable Hardware Abstraction Layer to ensure genericity while supporting target hardware of the Tulipp Reference Platform. In other words, we defined the hardware-to-OS interfaces of the Tulipp Reference Platform.

These concepts we developed for HIPPEROS in the framework of this task could be applied to any OS supporting the Tulipp Reference Platform. The work done in this work package (i.e. supporting all hardware-to-OS interface at the OS level) should be done by any RTOS provider that would like to comply with the Tulipp Reference Platform and be part of the Tulipp product ecosystem. We achieved the necessary work to allow the use of HIPPEROS as a compliant RTOS with the Tulipp Reference Platform as defined in deliverable D1.1. The HAL provides a convenient generic layer to efficiently add the support of the chosen target (and other future potential targets).

8. References

[1] Tulipp Proposal Submission Forms, Horizon 2020, Call H2020-ICT-2015, Proposal number 688403.

[2] ARM architecture, https://en.wikipedia.org/wiki/ARM_architecture

[3] Cortex-A9 Processor, https://www.arm.com/products/processors/cortex-a/cortex-a9.php

[4] EMC²-Z7015 board, http://www.sundance.technology/som-cariers/pc104-boards/emc%C2%B2-dp-carrier/

[5] Trenz Electronic TE0715 board, https://shop.trenz-electronic.de/en/Products/Trenz-Electronic/TE07XX-Zynq-SoC/TE0715-Zynq-SoC/

[6] Zynq-7000 All Programmable SoC, https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[7] QEMU, http://wiki.qemu.org/Main_Page

[8] Vivado Design Suite, https://www.xilinx.com/products/design-tools/vivado.html

[9] Zedboard, http://zedboard.org/

[10] OpenOCD, an Open On-Chip Debugger, http://openocd.org/