D3.6 User Guide of the Hypervisor

V1.0

Document information

Contract number	687902
Project website	www.safepower-project.eu
Contractual deadline	31/12/2017
Dissemination Level	PU
Nature	Other
Author	FEN
Contributors	IKL
Reviewer	IMP
Keywords	User Guide, XtratuM, ARM, Hypervisor, Power

Notice:

This project and the research leading to these results has received funding from the European Community's H2020 program [H2020-ICT-2015] under grant agreement 687902

Change log

VERSION	DESCRIPTION OF CHANGE
V0.1	First Draft FEN (Salva Peiró, Patricia Balbastre, Javier Coronel, Manuel Muñoz, Alfons Crespo)
V0.2	Second Draft FEN (Salva Peiró, Patricia Balbastre, Javier Coronel, Manuel Muñoz, Alfons Crespo)
V1.0	Final Version FEN (Salva Peiró, Patricia Balbastre, Javier Coronel, Manuel Muñoz, Alfons Crespo) after reviews from IMP and IKL.

Table of contents

Table of figures

1. EXECUTIVE SUMMARY

This document is the User Guide of the XtratuM Hypervisor power extensions developed in the framework of the SAFEPOWER project to develop efficient power aware partitioned applications.

2. PREFACE

In this chapter you will find information about the target audience, the typographical conventions followed in this document, as well, as this document's contact, support, and copyright information.

2.1. Audience

The target readers of this document are software developers using the services of the XtratuM hypervisor power extensions on XM-ARM. The reader is expected to have an in-depth knowledge of the ARM Cortex-A9 architecture, and, experience in the programming device drivers on the ZYNQ-7000 hardware platform. It is also advisable to have knowledge of the ARINC-653, and, other related standards.

2.2. Typographical conventions

The following typographical conventions are used in this document:

- typewriter: used in assembler and C code examples, and to show the output of commands.
- *italic face*: used to introduce new terms.
- **bold face**: used to emphasize or highlight a word or paragraph.

Code examples are printed inside a box as shown in the following example:

```
void PartitionMain(void)
{
    printf("Hello world!\n");
}
```

2.3. Support

Fent Innovative Software Solutions (FentISS)
Camino de vera s/n
CP: 46022
Valencia, Spain

The official XtratuM web site is: http://www.fentiss.com

3. INTRODUCTION

This document is the User Guide of the XtratuM Hypervisor power extensions. The contents of this user guide address how to use the XtratuM hypervisor extensions developed in the framework of the SAFEPOWER project to develop efficient power aware partitioned applications.

3.1. Scope and Objectives

The scope of the document is the User Guide of the XtratuM XM-ARM hypervisor [1]. The XtratuM hypervisor operates on the Processing System (PS) part of the ZYNQ-7000 [2]. Therefore, the main focus of this document is describing the services, interfaces, and, tools provided by the XM-ARM hypervisor for the development of power efficient XM-ARM partitioned applications running on the PS part (Cortex-A9 ARM cores) of the ZYNQ-7000 hardware platform [2].

The Programmable Logic (PL), eg. FPGA is not managed by the XtratuM XM-ARM hypervisor operation, as the hypervisor has no control over the software operating on the PL Part. However, the hypervisor provides services to the Cortex-A9 ARM partitions on the PS to communicate with the PL over the Network On Chip (NoC) [3].

3.2. Document structure

This document is structured in two main parts:

First, the Chapter XtratuM Hypervisor Tools discusses the overall architecture of the XtratuM hypervisor power extensions, and, its relation with the rest of the XtratuM Hypervisor components, namely: the xDRAL programming interface used by the partitions, the XSDK-XM drivers to program hardware devices, the Xoncrete scheduling tool obtain the Hypervisor cyclic scheduling plans.

Second, the Chapter Application Development guide provides a hands-on application development guide based around the implementation of use-cases of the safe and low power services provided by the XtratuM ARM hypervisor, such as: basic setup of the system, low power schedule profiles, Monitoring services, NoC communications, and, the Ethernet network communications.

4. XTRATUM HYPERVISOR TOOLS

The XtratuM Hypervisor is extended with the services, and tools developed in the framework of the SAFEPOWER project to enable the generation of safe and power aware partitioned applications. This chapter is in charge of introducing the XtratuM Hypervisor Tools available to partition developers, to that end, the current chapter is structured as follows: First, the section Overview of XtratuM Tools provides an overview of the XtratuM XM-ARM hypervisor tools. Then, the remaining sections of this chapter present each of the XtratuM hypervisor XM-ARM components. For further details about the architectural design of the XtratuM hypervisor power extensions, the reader is referred to the deliverable "D3.5 Architectural Design of the Hypervisor" [4].

4.1. Overview of XtratuM Tools

An overview of the XtratuM Hypervisor tools is depicted in the Figure (1). The following elements compose the architecture, starting from the bottom of the diagram:

- **ZYNQ-7000** The underlying ZYNQ-7000 Xilinx hardware platform [2].
- **XtratuM Hypervisor** The XtratuM Hypervisor ensures Temporal and Spatial Isolation (TSP), and, provides the safe low power techniques (LPT).
- **XMCF** The XtratuM Configuration file (XMCF) that defines the system.
- **xDRAL** Provides the interface used by the end-user applications to develop power aware partitioned applications.
- **XSDK-XM** Provides the Xilinx SDK API [5] drivers ported to the XtratuM XM-ARM hypervisor that enables using the existing XSDK drivers with XtratuM partitions [6].
- **Xoncrete** The Xoncrete scheduling tool [7] generates the XMCF configuration file with the power aware scheduling profiles.

4.2. XtratuM Hypervisor

The XtratuM ARM hypervisor is a software layer that creates and executes multiple virtual machines on a hardware target machine for high critical safety systems [8], [9], [10]. Each software system is executed in a virtual environment that mimics the behaviour of the underlying hardware architecture. The XtratuM ARM hypervisor enables the design of multi-core time and space partitioning (TSP) based systems.

4.2.1. XMCF

The XtratuM configuration file (XMCF) defines the static configuration of the system resources, namely, time and memory allocation to partitions, communication channels, and, access to hardware devices present on the system.

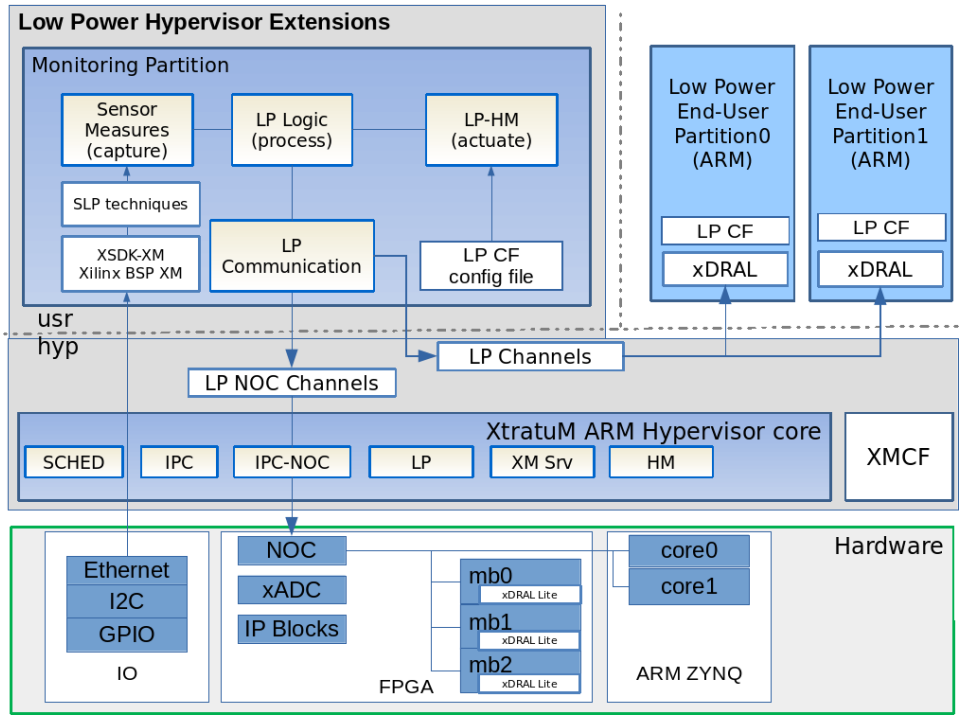

Figure 1: XtratuM hypervisor Low Power extensions detailed diagram

4.3. xDRAL

The Extended DRAL (xDRAL) abstraction layer [11] is a software component developed in the framework of the SAFEPOWER project that extends the DREAMS Abstraction Layer (DRAL) [12] providing power monitoring, and, power aware services to the XtratuM partitions executing on the Cortex-A9 ARM cores [13] of the ZYNQ-7000 hardware platforms.

4.3.1. Interface

The xDRAL API [11] is extended with the following power-aware services that are available to partition developers to implement efficient power aware applications:

- **XDRAL_SIGNAL_ACTIVITY_COMPLETION.** The XDRAL_SIGNAL_ACTIVITY_COMPLETION idles the execution of the calling partition. This service transfers control to the hypervisor that determines the power saving technique to apply.
- **XDRAL_GET_SENSORS_INFO.** Returns the low-power monitoring sensor channels information.
- **XDRAL_SET_MODULE_SCHEDULE.** The XDRAL_SET_MODULE_SCHEDULE requests for a schedule plan change. The schedule defines the slot frequency for each scheduling slot composing the schedule, effectively performing ARM CPU core

frequency scaling according to the slots frequency defined in the XMCF configuration.

- **XDRAL_GET_PROFILE.** The XDRAL_GET_PROFILE service [11] returns the (mode, profile) tuple corresponding to the current XMCF schedule as defined in the LPCF.
- **XDRAL_SET_PROFILE.** The XDRAL_SET_PROFILE service [11] set the XMCF schedule plan to the schedule corresponding to the (mode, profile) tuple defined in the LPCF.
- **XDRAL_GET_SLACK_TIME.** The XDRAL_GET_SLACK_TIME service returns the partition slack time accounted during the major Frame.
- **XDRAL_GET_FREQUENCY.** Return the operating frequency of the ARM CPU cores.

For further details about the operation of each xDRAL service the reader is referred to the "xDRAL: Software User Manual" [11].

4.3.2. Architecture

Presents xDRAL Architecture Overview

- **Monitoring Partition**: Extend the hypervisor solution with power sensor monitoring and Low Power Techniques (LPTs at the partition level.
- **End-user Partition**: The end-user partitions are the partitions where the end-user application payload are executed. The end-user applications interact with the rest of the system through the Extended DRAL (xDRAL) interface.
- **LPCF**: The Low Power Configuration File LPCF provides the static configuration of the LP resources.

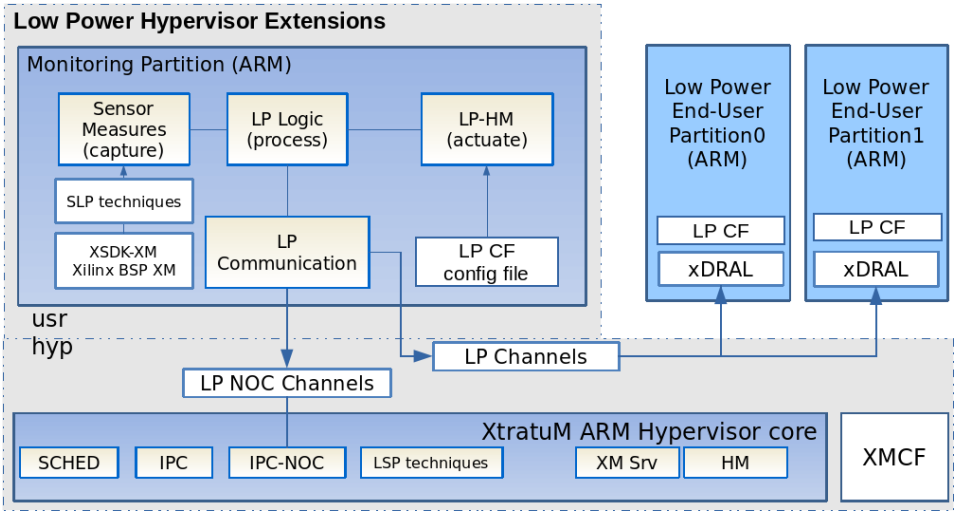

Figure 2: Architecture of XtratuM ARM hypervisor LP extensions

As an example, the sequence diagram in Figure (3) depicts the sequence of actions that take place when the XDRAL_GET_SENSORS_INFO() service is invoked from an end-user partition. Two partitions are involved in the sequence diagram: (1) The end-user partition requesting the XDRAL_GET_SENSORS_INFO() service, and, (2) The monitor partition in charge of providing up-to-date monitor sensor data. The actions carried by each of the involved partitions are detailed next.

The end-user partition invokes the XDRAL_GET_SENSORS_INFO() service:

1. The partition application invokes the XDRAL_GET_SENSORS_INFO() service of the xDRAL interface.

2. The XDRAL_GET_SENSORS_INFO() service implementation invokes the XM_read_sampling_port() service [14] to reads the sensor information from the sampling channel LP-Channel.

3. The XDRAL_GET_SENSORS_INFO() service implementation receives the sensor information when the XM_read_sampling_port() [14] returns.

4. The XDRAL_GET_SENSORS_INFO() service implementation returns the temperature sensor reading obtained from the LP-Channel to the application running on the end-user partition.

The monitoring partition periodically updates the sensor information available on the LP-Channel:

1. The monitoring partition reads the sensor information from the hardware, eg. invoking the I2C read() service provided by the Xilinx BSP drivers [6].

2. The monitoring partition writes the sensor information retrieved from the hardware to the sampling channel LP-Channel invoking XM_write_sampling_port() service [14].

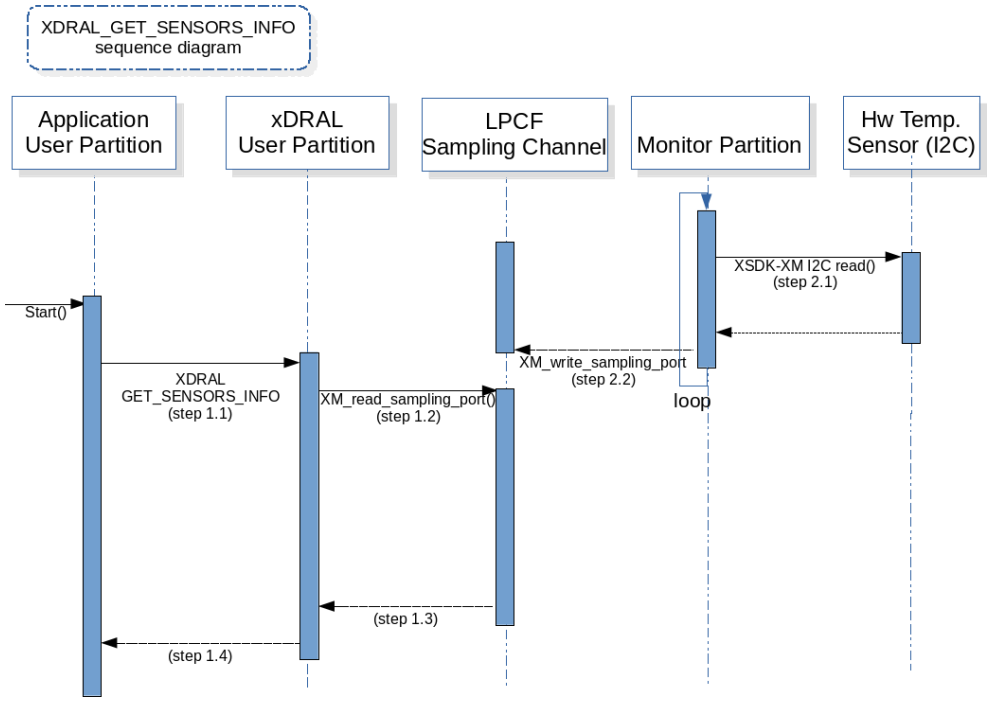

Figure 3: XDRAL_GET_SENSORS_INFO() service sequence diagram

4.3.3. LPCF

The Low Power Configuration File (LPCF) [11] defines the configuration of the Low-Power resources used by the xDRAL partitions in the framework of the SafePower project. The LPCF interface provides the following type definitions used by the xDRAL partitions:

- **LPCF Configuration Types** Defines the configuration of the Low Power resources, such as, the "LP-Channel" sampling channel, that communicates the Monitoring Partition with the end-user partitions. As well as, the LPCF Health Monitoring configuration containing the set of LP HM tuples (event, action) that defines the actions to be performed by the monitoring partition, when a LP HM event occurs.
- **LPCF Data Types** Defines the data types exchanged by the Monitoring partition and the end-user partitions.

For further details about the LPCF the reader is referred to the xDRAL Manual [11].

4.3.4. Safe Low Power (SLP)

The Safe Low Power (SLP) is a software component that is integrated into the monitoring partition to provide the implementation of the low power techniques (LPTs) [15] at the partition level. Although the SLP component is not part of the XtratuM hypervisor, the SLP component is presented here for completeness of the overall system architecture. For further details about the SLP the reader is referred to the D2.2 deliverable [15].

4.4. XSDK-XM

The Xilinx SDK API [5] is a Board Support Package (BSP) that provides full suite of libraries and device drivers on the ZYNQ-7000 platform. The availability of this libraries, and, device drivers eases the development of new bare metal applications on the ZYNQ-7000 hardware boards by providing an standard interface.

In order to re-use the drivers, and, ease the portability on the SafePower applications, FentISS has ported the XSDK Xilinx drivers API [5] to run on XtratuM XM-ARM partitions. As a result, the XSDK-XM BSP [1] has been developed by FentISS, together, with the accompanying libraries such as the lwIP library [16] that enable building TCP/IP networked applications, such, as the Ethernet communication application.

The XSDK-XM BSP is a key building block that provides access to the hardware devices, such as, the XADC, and, I2C, required by the Monitoring and Safe Low Power (SLP) services to implement the voltage monitoring, and, power scaling services. For further details about the XSDK-XM BSP [1], and, its associated documentation, please, refer to the fentiss.com/en/products/execution.html website.

4.5. Xoncrete

Xoncrete [7] is an analysis tool that performs the schedulability analysis of a partitioned system. Rater than a general purpose scheduling tool, Xoncrete has primarily been designed to meet the ARINC 653 system model. But it also allows to analyze other scheduling policies for processes such as EDF. Besides the scheduling analysis capabilities, the Xoncrete tool also provides a user friendly interface for capturing and editing all the elements that are part of a partitioned system. It has been specially designed to generate configuration files compatible with XtratuM.

In the framework of the SAFEPOWER project, the Xoncrete tool has been extended with scheduling algorithms and power profiles described in [4]. For further details about the Xoncrete tool, and, its associated documentation, please, refer to the fentiss.com/en/products/xoncrete.html website, that provides a hands on tutorial on using the main features of Xoncrete.

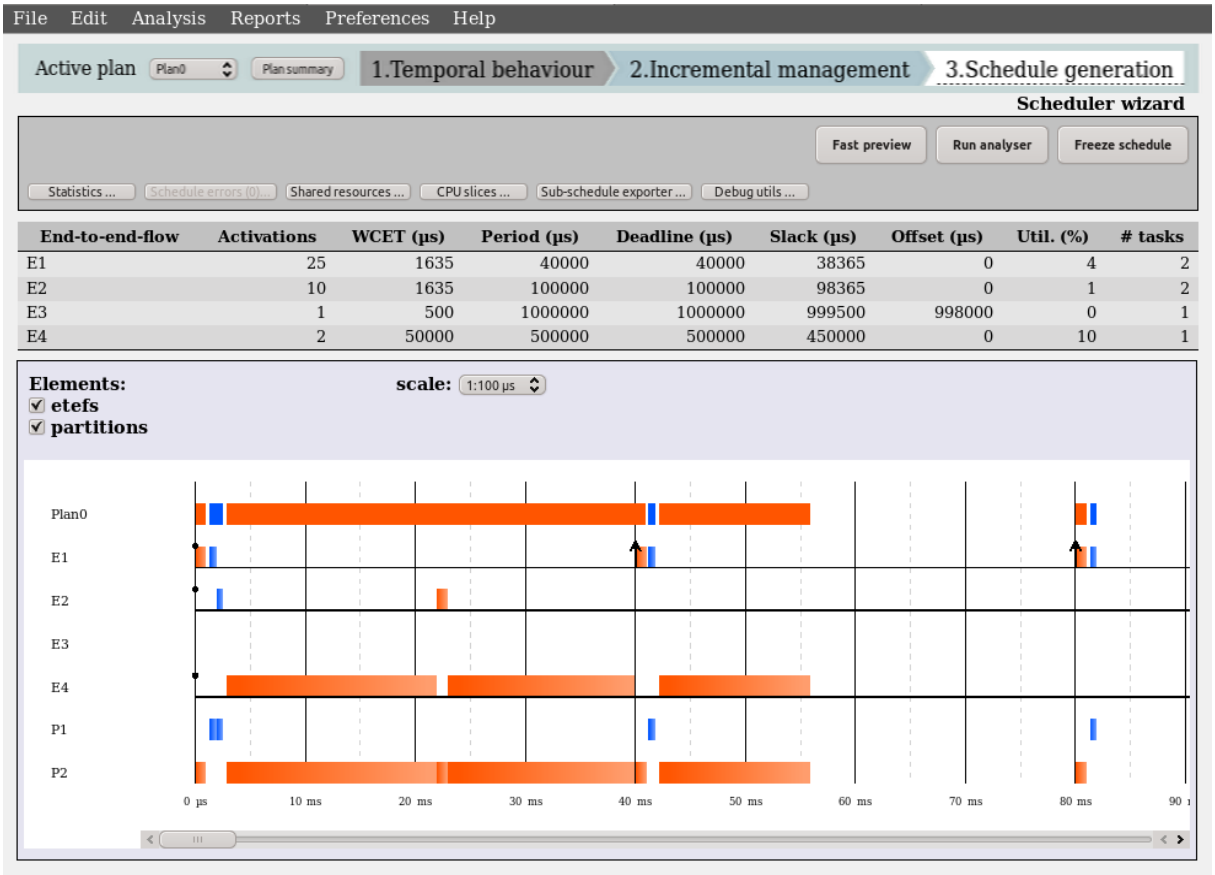

Figure 4: Xoncrete Analysis Interface screenshot

4.6. ZYNQ-7000

The ZYNQ-7000 represents the hardware board where the final end-user applications are executed. The ZYNQ-7000 hardware depicted in the Figure (1) is composed of the main parts described in ZYNQ-7000-TRM [2] with the extended PCB features detailed in D3.4 [17]:

- **PS Part**: Composed of the Cortex-A9 cores (PS) executing the XtratuM ARM hypervisor and end-user partitions.
- **PL Part**: Composed of the FPGA executing the MicroBlazes communicated through the Network on Chip (NoC).
- **I/O Peripherals**: The I/O peripherals present on the ZYNQ-7000 board, such as the Ethernet, I2C, USB described in [2].

5. APPLICATION DEVELOPMENT GUIDE

This chapter provides an application development guide focused on the development of applications using the safe, and, power aware services offered by the XtratuM Hypervisor Tools to implement safe and power efficient partitioned applications.

This chapter is structured as follows:

- First, the section Obtaining the XtratuM Hypervisor Tools, details the XM-ARM hypervisor tools required to develop the partitioned applications covered in this chapter.
- Next, the section Installing the XtratuM Hypervisor Tools details the setup of the XM-ARM hypervisor tools that compose the software development environment.
- Last, the section Developing the Application Examples covers how to develop and use the application examples in this chapter.

5.1. Obtaining the XtratuM Hypervisor Tools

The following list enumerates the XM-ARM hypervisor tools required to build and run the application examples in this guide, for each tool a brief description of the tool is provided, and, the URL location where the tool can be obtained.

- **XtratuM Hypervisor.** The XtratuM ARM Hypervisor Software Development Kit (SDK) to develop partitions on the ZYNQ-7000. The XtratuM hypervisor SDK can be obtained at fentiss.com/en/products/xtratum.html website.
- **xDRAL-ARM SDK.** The xDRAL is the interface used by the end-user applications to interact with the rest of the system. The xDRAL can be obtained at fentiss.com/en/products/execution.html website.
- **XSDK-XM ARM.** The XSDK Xilinx drivers API for the ZYNQ-7000 ported to the XM-ARM Hypervisor. The XSDK-XM ARM can be obtained at fentiss.com/en/products/execution.html website.
- **Xoncrete.** The Xoncrete offline scheduling analysis tool used to generate efficient cyclic schedule plans. The Xoncrete scheduling tool can be obtained at fentiss.com/en/products/editors.html website.

5.2. Installing the XtratuM Hypervisor Tools

Before beginning with the development application examples first, the XtratuM hypervisor development environment must be setup. The development environment is composed by the XtratuM Hypervisor Tools:

- **Setup of the XtratuM Hypervisor.** Please, refer to the document “Software Development Start Guide: XM-ARM” [18] that details the steps to install the

XtratuM hypervisor, the toolchain, required tools, and, the setup of the ZYNQ-7000 hardware development boards.

- **Setup of the xDRAL for XM-ARM.** Please, refer to the document "xDRAL Software User Manual" [11] that details the steps to install the xDRAL, and, its accompanying tools.
- **Setup of the XSDK-XM ARM.** Please, refer to the document "XSDK-XM Software Release document" [6] that details the steps to install the XSDK-XM, and, its accompanying tools.

5.3. Developing the Application Examples

The following application examples detail how to use the XM-ARM Hypervisor tools develop the following applications:

- **Hello world application** Getting started with the development tools by developing an partitioned hello-world.
- **Monitor partition application** Develop partitioned applications that access the monitoring services provided by the Monitor partition.
- **Power profile application** Develop partitioned applications using power modes/ scheduling profiles to reduce power consumption.
- **Ethernet communication application** Develop partitioned applications that communicate with a control workstation through Ethernet using the TCP/IP protocol.
- **NoC communication application** Develop partitioned applications that communicate through the internal Network On Chip (NoC) with the MicroBlazes running on the PL side.

5.3.1. Hello world application

The xDRAL hello world is a simple application composed of two xDRAL partitions that print the "Hello World" message to the ZYNQ-7000 UART.

5.3.1.1. Compilation steps

Compile the application example using the make [19] utility to produce the system image (resident_sw).

```
$ cd xdral/examples/hello_world
hello_world $ make clean all
hello_world $ ls resident_sw
resident_sw
```

- **Execution steps**

Use the Xilinx xsdb tool [20] to execute the resident_sw image.

```
hello-world$ xsdb
xsdb% connect
xsdb% rst -srst
xsdb% dow resident_sw
xsdb% run
xsdb% exit
```

5.3.1.2. Expected output

The application is composed of two XtratuM XM-ARM partitions that run on the Cortex-A9 CPU0 core. Therefore, upon startup each partition prints the “Hello world” messages to the ZYNQ-7000 UART as shown in the listing below.

```
XM Hypervisor (2.2 r0) Built Oct  4 2017 17:29:13
Detected 400.0MHz processor.
>> HWClocks [CortexA9 Global Clock (1000Khz)]
>> HwTimer [CortexA9 Private Timer1 (1000Khz)]
2 Partition(s) created
P0 ("Partition0":0) flags: [ SYSTEM FP ]:
    [0x10000000:0x10000000 - 0x1003ffff:0x1003ffff] flags: 0x0
P1 ("Partition1":1) flags: [ SYSTEM FP ]:
    [0x10040000:0x10040000 - 0x1007ffff:0x1007ffff] flags: 0x0
[P0] Hello World!
[P1] Hello World!
```

5.3.2. Monitor partition application

The Monitor partition application is an application composed of two xDRAL partitions:

(a) **Monitor partition** The monitor partition uses the I2C XSDK-XM drivers to access the power controllers available on the ZYNQ-7000. The power controllers are sampled periodically by the monitoring partition, the obtained power readings are provided to the End-user partition through the XDRAL_GET_SENSOR_INFO service described in the XDRAL User Manual [11].

(b) **End-user partition** The end-user partition uses the XDRAL_GET_SENSOR_INFO service [11] to retrieve the power readings.

The communication between the Monitor and the End-user partition takes place over a sampling port defined in the XtratuM Configuration File (XMCF). The Low Power Configuration File (LPCF) identifies the channel defined in the (XMCF), as the channel used to exchange of monitoring information between the Monitor and the End-user partitions.

In addition, the LPCF defines the LP HM configuration, with the LPCF (HM Event, HM Action) tuples. The LPHM action enable to specify actions taken when HM events are detected, such as upper/lower temperature/voltage/power thresholds reached. For further details about the LPCF configuration creation, please check the XDRAL User Manual [11].

5.3.2.1. Compilation steps

Compile the application example using the make [19] utility to produce the system image (resident_sw).

```
$ cd xdral/examples/monitoring_partition/
monitoring_partition $ make clean all XSDK_PATH=/path/to/xsdk-xm
monitoring_partition $ ls resident_sw
resident_sw
```

5.3.2.2. Execution steps

Use the Xilinx xsdb tool [20] to execute the resident_sw image.

```
monitoring_partition $ xsdb
xsdb% connect
xsdb% rst -srst
xsdb% dow resident_sw
xsdb% run
xsdb% exit
```

5.3.2.3. Expected output

The application is composed of three XtratuM XM-ARM partitions that run on the Cortex-A9 CPU0 core. The partition P0 is the Monitor partition, while the partitions P1, and, P2, are End-User partitions. The End-User partitions use the XDRAL_GET_SENSOR_INFO service [11] to retrieve the power readings, and, write the message “UP RECV N=X T=Y” messages to the ZYNQ-7000 UART as shown in the listing below.

During execution the temperature sensor reaches the upper threshold, then, the Monitor Partition detects the situation and applies the action defined in the LPCF HM section of the LPCF, in the example shown in the output below, the action configured is to switch to DegradedMode2, as it can be seen in the LPHM: lpcfHmSwitchToDegradedMode2 message in the output.

```
XM Hypervisor (2.2 r0) Built Dec 12 2017 18:02:06
Detected 400.0MHz processor.
>> HWClocks [CortexA9 Global Clock (1000Khz)]
>> HwTimer [CortexA9 Private Timer1 (1000Khz)]
3 Partition(s) created
P0 ("Partition0":0) flags: [ SYSTEM ]:
    [0x14000000:0x14000000 - 0x1407ffff:0x1407ffff] flags: 0x0
P1 ("Partition1":1) flags: [ ]:
```

```
    [0x18000000:0x18000000 - 0x1807ffff:0x1807ffff] flags: 0x0
P2 ("Partition2":2) flags: [ ]:
    [0x1c000000:0x1c000000 - 0x1c07ffff:0x1c07ffff] flags: 0x0
InitSecondaryCpu 1 0x21007140
>> HwTimer [CortexA9 Private Timer1 (1000Khz)]

[P0] MP OPEN lpSamplingPort at 4194304
[P2] UP RECV N=0 T=0 0=0
[P1] UP RECV N=0 T=0 0=0
[P0] MP SEND N=1 T=0 2=201
[P2] UP RECV N=1 T=1051292 2=201
[P1] UP RECV N=1 T=1051292 2=201
[P0] MP SEND N=1 T=1051292 3=202
[P2] UP RECV N=1 T=2051954 3=202
[P1] UP RECV N=1 T=2051954 3=202
[P0] MP SEND N=1 T=2051954 4=203
[P2] UP RECV N=1 T=3051962 4=203
[P1] UP RECV N=1 T=3051962 4=203
[P0] MP SEND N=1 T=3051962 5=204
[P0] LPHM: lpcfHmSwitchToDegradedMode2: 0
[P1] UP RECV N=1 T=4051959 5=204
[P2] UP RECV N=1 T=4051959 5=204
[P0] MP SEND N=1 T=4051959 6=205

[P0] DONE
```

5.3.3. Power profile application

The Power profile applications is composed of a partition that invokes the XDRAL_SET_PROFILE() service [11] to change between the power profiles defined in the LPCF. The LPCF defines the scheduling power profiles available when invoking XDRAL_SET_PROFILE. For further details about the LPCF configuration creation, please check the XDRAL User Manual [11].

5.3.3.1. Compilation steps

Compile the application example using the make [19] utility to produce the system image (resident_sw).

```
$ cd xdral/examples/profile_management
profile_management $ make clean all
profile_management $ ls resident_sw
resident_sw
```

5.3.3.2. Execution steps

Use the Xilinx xsdb tool [20] to execute the resident_sw image.

```
profile_management $ xsdb
xsdb% connect
xsdb% rst -srst
xsdb% dow resident_sw
xsdb% run
xsdb% exit
```

5.3.3.3. Expected output

```
XM Hypervisor (2.2 r0) Built Oct  4 2017 17:29:13
Detected 400.0MHz processor.
>> HWClocks [CortexA9 Global Clock (1000Khz)]
>> HwTimer [CortexA9 Private Timer1 (1000Khz)]
2 Partition(s) created
P0 ("Partition0":0) flags: [ SYSTEM FP ]:
    [0x10000000:0x10000000 - 0x1003ffff:0x1003ffff] flags: 0x0
P1 ("Partition1":1) flags: [ SYSTEM FP ]:
    [0x10040000:0x10040000 - 0x1007ffff:0x1007ffff] flags: 0x0
[P0] SET_PROFILE(0,0): ret 1
[P1] GET_PROFILE(0,0): ret 0
[P0] SET_PROFILE(1,0): ret 0
[P1] GET_PROFILE(0,0): ret 0
[P0] SET_PROFILE(1,1): ret 0
[P1] GET_PROFILE(1,0): ret 0
[P0] SET_PROFILE(2,0): ret 0
[P1] GET_PROFILE(1,1): ret 0
[P0] SET_PARTITION_MODE(HALT)
```

5.3.4. Ethernet communication application

The Ethernet communication application is a partitioned system that uses the Ethernet Controller present on the ZYNQ-7000. The example provides a partition that uses the XSDK-XM drivers to implement an application that communicates through Ethernet. The example can be tailored by the end-user, eg. to configure the MAC and/or TCP/IP layer addresses, and, to implement the specific application communication.

The XSDK-XM [5] provides the XSDK drivers and lwIP network stack ported to XtratuM ARM [1]:

1) The EMACP driver for the GEM0 Ethernet device present in the ZYNQ-7000 [2].

2) The lwIP TCP/IP network stack that enables the transmission and reception of TCP/IP packets [21].

5.3.4.1. Compilation steps

Compile the application example using the make [19] utility to produce the system image (resident_sw).

```
# Compile the XSDK-XM Board Support Package (BSP) for XM-ARM
make -C xsdk-xm/xsdk_xmarm_bsp

# Compile the XSDK-XM Ethernet application example
make -C xsdk-xm/examples/xsdk-ethernet-example/Build/
```

5.3.4.2. Execution steps

Use the Xilinx xsdb tool [20] to execute the resident_sw image. The xsdb tool performs the steps detailed in the xsdb.ini provided with the application example:

1) Initialise the ZYNQ-7000 according to the ps7_init.tcl initialisation script.

2) Load the resident_sw image and execute it.

```
workstation ~ $ cd xsdk-xm/xsdk-ethernet-example/Build/
workstation Build $ xsdb
```

3) Setup the network interface (eth1) on the workstation. Then, use the telnet command in the workstation to connect to the echo server present on the xsdk-ethernet-example application running on the ZYNQ-7000.

```
# Connect to ZYNQ-7000 10.0.0.2 from the workstation host at 10.0.0.1
workstation ~ $ sudo ifconfig eth1 10.0.0.1
workstation ~ $ telnet 10.0.0.2
```

5.3.4.3. Expected output

The example is composed by one XtratuM XM-ARM partition running on the Cortex-A9 CPU0 core that implements a TCP echo server running on IP 10.0.0.2. Therefore, the expected output is observing on the UART the sequence messages echoed by the TCP echo server as shown in the listing below.

```
XM Hypervisor (2.2 r0) Built Oct 31 2017 15:33:53
Detected 400.0MHz processor.
>> HWClocks [CortexA9 Global Clock (1000Khz)]
>> HwTimer [CortexA9 Private Timer1 (1000Khz)]
2 Partition(s) created
P0 ("Partition0":0) flags: [ ]:
    [0x10000000:0x10000000 - 0x10effffff:0x10effffff] flags: 0x0
    [0x10f00000:0x10f00000 - 0x10ffffff:0x10ffffff] flags: 0x4
    [0x11000000:0x11000000 - 0x110fffff:0x110fffff] flags: 0x0
    [0xe000b000:0xe000b000 - 0xe000bfff:0xe000bfff] flags: 0x0
P1 ("Partition1":1) flags: [ SYSTEM FP ]:
    [0x12000000:0x12000000 - 0x120fffff:0x120fffff] flags: 0x0

[P0] -----lwIP RAW Mode Demo Application ------
[P0] Board IP:       10.0.0.2
```

```
[P0] Netmask :       255.255.255.0
[P0] Gateway :       10.0.0.1
[P0] Adding network interface to lwip
[P0] Start PHY autonegotiation
[P0] Waiting for PHY to complete autonegotiation.
[P0]
[P0] Auto negotiation error
[P0] autonegotiation complete
[P0] link speed: 100
[P0] Xemac Handler attached to IRQ 54
[P0] Added
[P0] Setting default: OK
[P0] Set up netif: OK
[P0] Init Timer
[P0] Enabled Interrupts
[P0] Start applications:OK
[P0]
[P0] Server          Port    Connect With
[P0] --------------  ------  --------------------
[P0]   echo server        7  $ telnet <board_ip> 7
[P0] rxperf server     5001  $ iperf -c <board ip> -i 5 -t 100
[P0] txperf client      N/A  $ iperf -s -i 5 -w 64k (on host with IP
192.168.1.100)
[P0]   tftp server       69  $ tftp -i 192.168.1.10 PUT <source-file>
[P0]   http server       80  Point your web browser to http://192.168.1.10
```

5.3.5. NoC communication application

The NOC communication example is composed of a XtratuM partition that communicates through the TTEL-NoC: The partition0 reads the data sent by the MicroBlazes running on the PL over the TTEL NoC using the XDRAL_READ_SAMPLING_MESSAGE() service [11]. The LPCF configuration file together with the XMCF defines the TTEL-NoC communication channels, that are available to the partition, in this scenario two Sampling ports are provided:

(1) **MB0_0_CA9_6**: The XtratuM Sampling port MB0_0_CA9_6 connects the MB0 microblaze port 0 with the Cortex-A9 port 6.

(2) **MB0_2_CA9_8**: The XtratuM Sampling port MB0_2_CA9_8 connects the MB0 microblaze port 2 with the Cortex-A9 port 8.

5.3.5.1. Compilation steps

Compile the application example using the make [19] utility to produce the system image (resident_sw).

```
$ cd xdral/examples/ttel_noc_communication/
ttel_noc_communication $ make clean all
ttel_noc_communication $ ls resident_sw
resident_sw
```

5.3.5.2. Execution steps

1) Copy the TTEL-NoC bitstream .bit file from the SAFEPOWER share-point into the folder “launch_script/”, eg. launch_script/Zero_release_20170310_USI.bit.

2) Execute the example on the ZYNQ-7000 board. Use the Xilinx xsdb tool [20] to execute the resident_sw image.

```
# The xsdb.ini file provided with the example runs the application.
ttel_noc_communication $ xsdb xsdb.ini
```

5.3.5.3. Expected output

The example is composed by one XtratuM XM-ARM partition running on the Cortex-A9 CPU0 core that displays on the UART the messages written to the NoC by the Microblaze MB0. Therefore, the expected output is observing on the UART a sequence of NoC read_messages operations as shown in the listing below.

```
XM Hypervisor (2.2 r0) Built Oct  4 2017 17:29:13
Detected 400.0MHz processor.
>> HWClocks [CortexA9 Global Clock (1000Khz)]
>> HwTimer [CortexA9 Private Timer1 (1000Khz)]
2 Partition(s) created
P0 ("sdp_0":0) flags: [ SYSTEM ]:
    [0x10000000:0x10000000 - 0x100fffff:0x100fffff] flags: 0x0
P1 ("sdp_1":1) flags: [ SYSTEM ]:
    [0x18000000:0x18000000 - 0x1807ffff:0x1807ffff] flags: 0x0

[P0] -------------------------------------
[P0] -- Starting SafePower TTEL-NOC TEST --
[P0] -------------------------------------
[P0] NoC OPEN MB0_0_CA9_6
[P0] NoC OPEN MB0_2_CA9_8
...
[P1] 19 read_message MB0_0_CA9_6 r 16 noc-data 0x18
[P1] 20 read_message MB0_2_CA9_8 r 16 noc-data 0x56
[P1]
[P1] 21 read_message MB0_0_CA9_6 r 16 noc-data 0x20
[P1] 22 read_message MB0_2_CA9_8 r 16 noc-data 0x66
...
```

6. TERMS, DEFINITIONS AND ABBREVIATED TERMS

6.1. Terms and definitions

- **Hypervisor** The layer of software that, using the native Cortex-A9 ARM hardware resources, provides one or more virtual machines (partitions).
- **Native hypervisor** Hypervisor that runs directly on the host's hardware Cortex-A9 ARM to control the hardware and to manage guest operating systems.
- **Partition** Also known as "virtual machine" or "domain". It refers to the environment created by the hypervisor to execute user code on the Cortex-A9 ARM cores.

6.2. Abbreviated terms

- **API:** Application Programming Interface.
- **DRAL:** DREAMS Abstraction Layer [12].
- **DRNoC:** DREAMS Network on Chip [12].
- **FPGA:** Field-Programmable Gate Array.
- **HM:** Health Monitor.
- **IPC:** Inter Partition Communication.
- **LP:** Low Power.
- **LWIP:** A Lightweight TCP/IP stack [21]
- **MMU:** Memory Management Unit.
- **MPU:** Memory Protection Unit.
- **NI:** Network Interface.
- **NoC:** Network on Chip.
- **PLL:** Phase-locked loop.
- **PL:** Programmable Logic [2].
- **PM:** Power Management.
- **PS:** Processing System [2].
- **RSW:** Resident Software (resident sw) [1].
- **SDK:** Software Development Kit.
- **SLP:** Safe-Low Power services [22].
- **SPNoC:** SAFEPOWER Network on Chip [22].
- **TSP:** Temporal and Spatial Partitioning [1].

- **TTEL:** Time-Triggered Extension Layer [3].
- **xDRAL:** Extended DRAL [11].
- **XMCF:** XtratuM Configuration File [1].
- **XSDK:** Xilinx SDK Standalone Device and Drivers API [5].
- **XSDK-XM:** Xilinx SDK ported to XtratuM ARM Hypervisor [6].

7. BIBLIOGRAPHY

[1] Fent Innovative Software Solutions, “Software User Manual: XM-ARM,” Fent Innovative Software Solutions, S.L., 15-007.009.sum.03, Dec. 2017.

[2] Xilinx, “Zynq-7000 All Programmable SoC Technical Reference Manual (UG585),” 2015.

[3] University of Siegen, “TTEL Software Extensions (ver 1),” SAFEPOWER Consortium, Feb. 2017.

[4] SAFEPOWER, “D3.5: Architectural Design of the Hypervisor,” SAFEPOWER Consortium, Confidential D3.5, Jun. 2017.

[5] Xilinx, “Xilinx SDK Standalone Device and Drivers API,” 2015. [Online]. Available: https://www.xilinx.com/products/design-tools/embedded-software/sdk.html. [Accessed: 23-Jun-2015].

[6] Fent Innovative Software Solutions, “XSDK-XM - Software Release Document,” Fent Innovative Software Solutions, S.L., 15-007.035.sreld.03, Dec. 2017.

[7] V. Brocal, M. Masmano, I. Ripoll, A. Crespo, P. Balbastre, and J.-J. Metge, “Xoncrete: A scheduling tool for partitioned real-time systems,” *Embedded Real-Time Software and Systems*, 2010.

[8] A. Larrucea, J. Perez, I. Agirre, V. Brocal, and R. Obermaisser, “A Modular Safety Case for an IEC-61508 Compliant Generic Hypervisor,” in *2015 Euromicro Conference on Digital System Design*, 2015, pp. 571–574.

[9] S. Trujillo, A. Crespo, A. Alonso, and J. Pérez, “MultiPARTES: Multi-core partitioning and virtualization for easing the certification of mixed-criticality systems,” *Microprocessors and Microsystems*, vol. 38, no. 8, pp. 921–932, 2014.

[10] E. Carrascosa, J. Coronel, M. Masmano, P. Balbastre, and A. Crespo, “XtratuM hypervisor redesign for LEON4 multicore processor,” *ACM SIGBED Review*, vol. 11, no. 2, pp. 27–31, 2014.

[11] Fent Innovative Software Solutions, “xDRAL - Software User Manual,” Fent Innovative Software Solutions, S.L., 15-007.015.sum.03, Dec. 2017.

[12] DREAMS, “ D2.3.1 XtratuM support of enhanced hypervisor layer services,” DREAMS: Distributed Real-time Architecture for Mixed Criticality Systems, D2.3.1, Sep. 2013.

[13] ARM, “Cortex-A9 Technical Reference Manual,” ARM Limited, ARM DDI 0406C.c, 2012.

[14] Fent Innovative Software Solutions, “Software Reference Manual: XM-ARM,” Fent Innovative Software Solutions, S.L., 15-007.010.sum.03, Dec. 2017.

[15] SAFEPOWER, “D2.2 Initial Low Power Techniques,” SAFEPOWER Consortium, D2.2, Dec. 2016.

[16] A. Dunkels, “Design and Implementation of the lwIP TCP/IP Stack,” 2001.

[17] SAFEPOWER, “D3.4 User guide of the PCB,” SAFEPOWER Consortium, Public D3.4, Jan. 2017.

[18] Fent Innovative Software Solutions, “Software Development Starter Guide: XM-ARM,” Fent Innovative Software Solutions, S.L., 15-007.011.sum.03, Dec. 2017.

[19] R. M. Stallman and R. McGrath, “GNU Make: A Program for Directed Recompilation, Version 3. 79. 1,” Free Software Foundation, 2002.

[20] I. Xilinx, “Embedded System Tools Reference Manual (UG1043),” Oct. 2016.

[21] Xilinx, “Xilinx XAPP1026 LightWeight IP (lwIP) Application Examples, v5.1, Application Note,” 2014. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf. [Accessed: 23-Jun-2015].

[22] SAFEPOWER, “D2.1 Definition of Reference Architecture.” SAFEPOWER Consortium, Public D2.1, Dec. 2016.

[23] SAFEPOWER Consortium, “SAFEPOWER Donwload Area,” 2017. [Online]. Available: http://safepower-project.eu/login/?redirect_to=http://safepower-project.eu/downloads/. [Accessed: 19-Dec-2017].

8. APPENDIX: XDRAL SPECIFICATION

The complete xDRAL specification is provided as an annexed document in the “xDRAL - Software User Manual” [8].

9. APPENDIX: SLP SPECIFICATION

The complete SLP specification is provided as an annexed document that will be available in the SAFEPOWER site, under the downloads section [23].