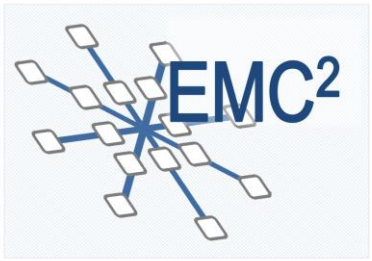

Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	**D4.6 – Specification of preliminary architecture and API respecting the requirements for error handling and redundancy of accelerators**	
Work package	WP4	Multi-core Hardware Architectures and Concepts
Task / Use Case	T4.4	Increased availability and dynamic reconfiguration
Subtasks involved		
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto:werner.weber@infineon.com	
Deliverable responsible	UTIA Jiri Kadlec, kadlec@utia.cas.cz, + 420 2 6605 2216	
Version number	v1.0	
Date	31/03/2015	
Status	Final	
Dissemination level	Public (PU)	

Authors

Partici-pant no.	Part. short name	Author name	Chapter(s)
04D	UTIA	Jiri Kadlec, Zdenek Pohl	Dynamic Reconfiguration of Asymmetric Multiprocessing Platform
15E	Tecnalia	Asier Alonso Munos, Daniel Mugica	Reliable and Self-Healing Dynamic Reconfiguration Manager (DRM) on LADAP platform (Virtex 5) (Cooperation with TASE)
17B	imec-nl	Tobias Gemmeke	Reliability Analysis in Presence of Hardware-induced Errors
16A	Chalmers	Ioannis Sourdis	Resilient Reconfigurable Multiprocessor arrays: probabilistic analysis of availability and efficiency

Document History

Version	Date	Author name	Reason
v0.1	14/01/2015	Zdenek Pohl	Initial Template, UTIA contribution draft
v0.2	02/03/2015	Jiri Kadlec	Integration of chapters: UTIA, Tecnalia, Imec-nl, Chalmers
v0.3	11/03/2015	Jiri Kadlec	Introduction and Conclusions
v0.4	16/03/2015	Jiri Kadlec	Ready for internal review in WP4
v0.5	31/03/2015	Jiri Kadlec	Implemented corrections requested by the internal project review. Text sent for second internal review.
v1.0	31/03/2015	Jiri Kadlec	Ready to be submitted to the ARTEMIS JU
	02/04/2015	Alfred Hoess	Final editing and formatting; deliverable submission

Publishable Executive Summary

This deliverable describes preliminary solution for the dynamically reconfigurable subsystems which can be eventually used in living labs/demonstrators. It describes reconfigurable architecture in several platforms, specific to the participating partners.

- Dynamic Reconfiguration of Asymmetric Multiprocessing Platform
- Reliable and Self-Healing Dynamic Reconfiguration Manager (DRM) on LADAP platform
- Reliability Analysis in Presence of Hardware-induced Errors
- Resilient Reconfigurable Multiprocessor arrays: probabilistic analysis of availability and efficiency

Specification of the preliminary architecture and API respecting the requirements for error handling and redundancy of accelerators outlines the intended extensions of the EMC^2 architecture by the support for reconfiguration of the hardware structures.

Table of contents

List of figures

List of tables

1. Introduction

This deliverable describes preliminary solutions for the dynamically reconfigurable subsystems which can be eventually used in living labs/demonstrators. It describes the planned reconfigurable architecture structures considered for the development within the EMC2 WP4.

First chapter is contributing with 2 forms of reconfiguration of HW as part of the Asymmetric Multiprocessing on the 28nm ZYNQ device. Floating point accelerators are reconfigured in the run-time by change of the firmware of internal scheduling controllers. The second level of dynamic reconfiguration is relying on the ZYNQ dual-core ARM processor. This processor is performing the dynamic reconfigurations of the complete programmable logic part of ZYNQ devices. UTIA is restricting the planned development to the dynamic reconfiguration of the complete programmable logic section of the ZYMQ device. No special (and expensive for the industrial partners) extensions of the design flow are needed for the proposed implementation of the dynamic reconfiguration of the complete programmable logic part of the ZYNQ. This can help adopting of the proposed technique. Main contributor is UTIA.

Second chapter is devoted to the implementation of complete Dynamic Partial Reconfiguration (DPR) for FPGA for the Space domain. A platform will be built which allows dynamic reconfiguration of Xilinx Virtex 5 FPGA with support of reliability and self-healing related features. It will take into account these issues, in the context of subtasks T4.3 and T4.4 and in relation to the living lab 3 (Space applications). Main contributor is Tecnalia.

Third chapter is addressing the required level of redundancy, and approaches how the underlying system has to be analysed with respect to its reliability in a systematic fashion. Determining the amount of required redundancy as well as to specify which type of errors are visible on software level, a connection has to be built between the occurrence of hardware errors and how they propagate through the system. The mitigation of errors is also addressed in this chapter. Cross layer approaches are proposed as most promising to achieve efficient trade-offs. Such cross layer approaches combine monitoring, control and correction mechanisms on multiple layers in the design. Main contributor is IMEC-NL.

The fourth chapter is proposing a probabilistic analysis to evaluate the availability of the Reconfigurable Multicore architecture designed in T4.2. This microarchitecture is formed by a reconfigurable processor array which is able to provide coarse-grain and fine-grain reconfigurability to mitigate permanent faults. Coarse grain reconfigurability will allow replacement of faulty processor parts by use of identical (presumably spare parts) borrowed by a neighboring processor. Fine-grain reconfigurability will provide further resources to install more processor parts to replace faulty processor components, acting as a wild card. Main contributor is Chalmers.

2. Dynamic Reconfiguration on ZYNQ Platform

Two forms of dynamic reconfiguration of HW are proposed to be developed for the Asymmetric Multiprocessing on the 28nm ZYNQ device.

- Floating point accelerators are reconfigured in the run-time by changing the firmware of internal scheduling controllers.
- The second level of dynamic reconfiguration relying on the ZYNQ dual-core ARM processor. This concept is not restricted only to the ARM core and ZYNQ platform. It relays on any hard-coded processor, which can continue to execute its program while the programmable logic is reconfigured.

This processor performs the dynamic reconfigurations of the complete programmable logic part of ZYNQ device. We propose to restrict the planned development to the dynamic reconfiguration of the complete programmable logic section of the ZYMQ device. No special (and expensive for the industrial partners) extensions of the design flow are needed for the proposed implementation of the dynamic reconfiguration of the complete programmable logic part of the ZYNQ. This can help to adopt the proposed technique.

The approach proposed has also its limitations. The ZYNQ device has to be used and connected to the external environment via its permanently running ARM dual core processing system part and I/O subsystem. The programmable logic part has to serve as an internal coprocessor connected only internally to the processing system part of the device. Main contributor is UTIA.

The dynamically reconfigurable ZYNQ platform will be also supporting the asymmetric multiprocessing [1]. It has this structure:

- ARM Cortex A9 dual core hard processor core (using ¾ of 1GB DDR3)
- Microblaze FPGA processor core (using ¼ of 1GB DDR3)
- Several EdkDSP accelerators [16], [17], [18], [19] each with 8x SIMD floating point data paths, reprogrammable by change of firmware of the PicoBlaze6 8bit controller. Firmware can be compiled by UTIA C compiler.

This chapter focuses on and is limited to the reconfiguration capabilities of the platform. Those interested in other details can use EMC2 D4.15, section “Demonstrator 2: Asymmetric Multiprocessing on ZYNQ” as a starting point.

The dynamically reconfigurable ZYNQ platform architecture will implement two different approaches to the reconfiguration.

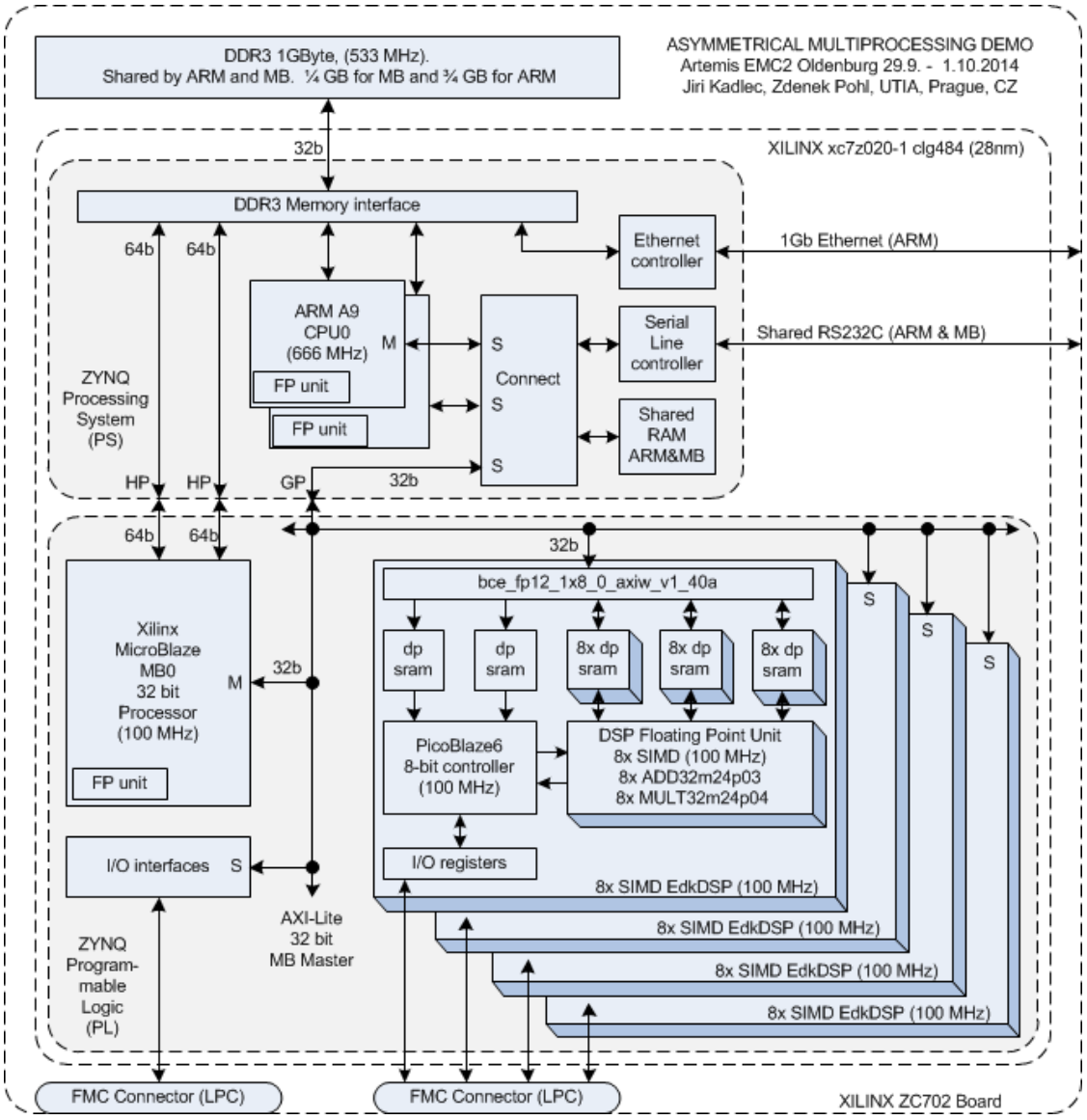

Figure 1: AMP demonstrator on ZYNQ

2.1 Reconfiguration by runtime reprogramming of firmware

Figure 2 shows the EdkDSP accelerators which are consisting of the DSP floating point unit for data processing and of PicoBlaze6 8bit microcontroller which controls the DSP unit operation. There are two PicoBlaze6 firmware memories (see 'dp sram' blocks connected to PicoBlaze6 in Figure 2), which hold the control code implementing complex DSP operations from elementary operations supported by the DSP unit. Thus the reconfiguration by firmware reprogramming is done by reprogramming of firmware memories. This way, the DSP operation implemented by the hardware accelerator is changed.

Unlike the fine grain reconfiguration of the FPGA, where individual bits of programmable logic configuration are changed, the reconfiguration of the firmware is limited to use 'switches and knobs' already prepared between high level blocks implemented in FPGA. For that reason, this kind of reconfiguration is sometimes denoted as coarse grained reconfiguration. The main advantage of such approach is in its high speed and reconfiguration efficiency, i.e. by changing single bit in one clock cycle the function of whole design/accelerator may change dramatically. On the other hand, the reconfiguration possibilities are limited only to those allowed by clever FPGA design done in advance.

2.1.1 **Results reached in month M12 of the EMC2 project**

UTIA has implemented two evaluation packages with the asymmetric multiprocessing design (AMP) [18] and [19]. Both packages are been based on the Xilinx application note XAPP1093 [1]. The AMP design [18] has been first ported from the ISE 14.5 design flow. The evaluation design [19] is already ported to to the Xilinx Vivado 2013.4 HW design flow with the Xilinx SDK 2013.4 SW design flow.

The ARM Cortex A9 processor works together with the Microblaze processor, sharing the terminal and block ram. Both processors execute program from the same external DDR3 memory. The Microblaze processor is controlling 4 EdkDSP floating point accelerators. Each accelerator is organised as 8xSIMD reconfigurable data path, controlled by the PicoBlaze6 controller. This evaluation package is provided by UTIA for the Xilinx ZC702 designs with AXI bus. Two released application notes [16] and [17] explain how to install and use the demonstrator on Windows7, (32 or 64 bit) and the Xilinx ZC702 board [2].

The evaluation package is demonstrated these parameters of the AMP system:

- Implementation of adaptive acoustic noise cancellation on 1 of 4 accelerators is computing the recursive adaptive LMS algorithm for identification of regression filter with 2000 coefficients in single precision floating point arithmetic with sustained performance
    - 632 MFLOP/s on the 100 MHz EdkDSP
    - 146 MFLOP/s on the 666 MHz ARM Cortex A9 (with the vector floating point unit)
    - 8 MFLOP/s on the 100 MHz Microblaze processor with the floating point HW unit
- The EdkDSP accelerators can be reprogrammed by the firmware. The programming is possible in C with the use of the UTIA EDKDSP C compiler. Accelerators can be programmed with two firmware programs. Designs can swap in the real time the firmware in only few clock cycles in the runtime.
- The alternative firmware can be downloaded to the EdkDSP accelerators in parallel with the execution of the current firmware. This is demonstrated by swap of the firmware for the FIR filter room response to the firmware for adaptive LMS identification of the filter coefficients in the acoustic noise cancellation demo.
- The EdkDSP accelerator is providing single-precision floating point results bit-exact identical to the reference software implementation running on Microblaze with the Xilinx HW single precision floating point unit.
- The 100 MHz 8xSIMD EdkDSP accelerator is 4,3x faster than the 666 MHz ARM Cortex A9 (with the vector processing unit) and 79x faster than computation on performance optimized 100 MHz Microblaze with HW floating point unit, in the presented case of the 2000 tap adaptive LMS filter.
- The floating point 2000 tap coefficients FIR filter (acoustics room model) is computed by single 100 MHz (8xSIMD) EdkDSP accelerator with the floating point performance 1007 MFLOP/s. The peak performance (only theoretical) of the single 100 MHz (8xSIMD) EdkDSP accelerator is 1,6 GFLOP/s.
- The peak performance of four 100 MHz (8xSIMD) EdkDSP accelerators implemented in this demo design is 6,4 GFLOP/s (this is only theoretical, peek figure).
- This evaluation package presents two (8xSIMD) EdkDSP accelerator families: one family without pipelined floating point divider data path and one family with a single pipelined floating point divider data path. The members of both families differ by size and by supported vector floating point operations.
- The floating point applications can be scheduled inside of the EdkDSP accelerator by the Xilinx PicoBlaze6 processor. Each firmware program has maximal size of 4096 (18 bit wide words).

2.1.2 Content of the released evaluation package

The asymmetric multiprocessing on ZYNQ (AMP) with the EdkDSP platform evaluation package [19] contains these deliverables for the Windows 7 (32 or 64bit):

- 8 evaluation versions of AMP designs. Each design contains one used ARM Cortex A9 processor core, one Microblaze and four instances of the EdkDSP accelerators with 8xSIMD floating point data paths with AXI-lite bus. (ARM 666 MHz, Microblaze 100 MHz, Accelerators 100 MHz) Designs are compiled in Xilinx Vivado 2013.4. See Figure 2.
- UTIA is providing source code for the demo applications and SW projects for the Xilinx SDK 2013.4 [19]. These source code projects are compiled with the UTIA library libwal.a serving for the EdkDSP communication.
- The included evaluation versions of the UTIA EdkDSP accelerators have HW limitation of maximal number of performed vector operations.
- The UTIA EdkDSPC C compiler is provided as 3 executable applications for Ubuntu in the VMware Player.
- The firmware is also provided in format of binary files to enable testing of accelerators without C compiler.

Partners of the Artemis EMC2 project can (after the signature of the zero cost contract with UTIA) get from UTIA (in addition to the precompiled evaluation designs) also the source code of Vivado 2013.4 HW design projects with the evaluation versions of the EdkDSP accelerators (in the Vivado 2013.4 IP netlist format). See Figure 2. See chapter 6 of the application note [19] for the detailed specification of the deliverables offered for the EMC2 project partners.

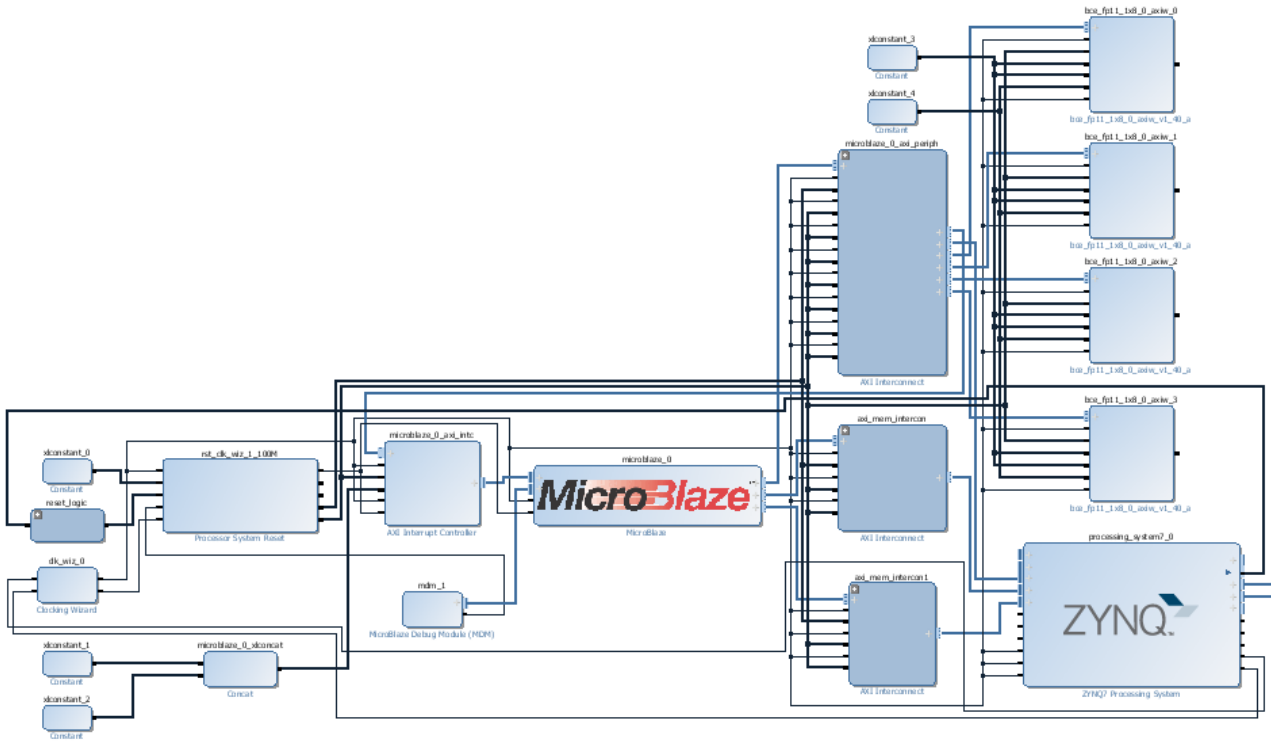

Figure 2: AMP evaluation design with 4 EdkDSP accelerators in Vivado2013.4 IP Integrator

The release versions of the AMP designs with the EdkDSP package for the Xilinx ZC702 board are also offered by UTIA. All customers can order from UTIA the release version of the AMP demo. It includes the Vivado 2013.4 HW design projects with the EdkDSP accelerators (in the Vivado 2013.4 IP netlist format) with main limitations removed. See sections 7 of the application note [19] for the specification of deliverables and license details.

2.1.3 **Description of installation and use of released evaluation designs**

The detailed UTIA application note [17] describes how to set-up and use of 8 HW designs running in an asymmetric multiprocessing architecture formed by of one ARM processor and one Microblaze processor with four (8xSIMD) EdkDSP accelerators on Xilinx ZC702 board. See Figure 3. The demonstrator serves for evaluation of parameters of these two UTIA floating point accelerator families in the AMP architecture on the Xilinx ZYNQ xc7z020-1 part:

- **bce_fp11_1x8_0_axiw_v1_[10|20|30|40]_a** is a family of four versions of floating point EdkDSP accelerators with 8 SIMD data paths.
- **bce_fp12_1x8_0_axiw_v1_[10|20|30|40]_a** is similar family of four versions of floating point EdkDSP accelerators with 8 SIMD data paths extended by a pipelined floating point division (FPDIV) in a single data path.

The four grades [10|20|30|40] of the EdkDSP accelerator differ in HW-supported vector computing capabilities:

The area optimized accelerators **bce_fp11_1x8_0_axiw_v1_10_a** and **bce_fp12_1x8_0_axiw_v1_10_a** perform vector floating point operations FPADD, FPSUB in 8 SIMD data paths.

The accelerators **bce_fp11_1x8_0_axiw_v1_20_a** and **bce_fp12_1x8_0_axiw_v1_20_a** perform vector floating point operations FPADD, FPSUB in 8 SIMD data paths plus the vector floating point MAC operations in 8 SMD data paths for length of the vector 1 up to 10. These accelerators can be used in applications like floating point matrix multiplication with row and column dimensions <= 10.

The accelerators **bce_fp11_1x8_0_axiw_v1_30_a** and **bce_fp12_1x8_0_axiw_v1_30_a** support identical operations as the bce_fp11_1x8_0_axiw_v1_20_a and bce_fp12_1x8_0_axiw_v1_20_a plus the floating point vector by vector dot products performed in 8 SIMD data paths. It is optimized for parallel computation of up to 8 FIR or LMS filters, each with size up to 255 coefficients. It is also effective in case of floating point matrix by matrix multiplications, where one of the dimensions is large (in the range from 11 to 255).

Finally, the accelerators **bce_fp11_1x8_0_axiw_v1_40_a** and **bce_fp12_1x8_0_axiw_v1_40_a** support identical operations as the bce_fp11_1x8_0_axiw_v1_30_a and bce_fp12_1x8_0_axiw_v1_30_a plus an additional HW support of dot product. It is computed in 8 data paths with the HW supported wind-up into single scalar result.

The **bce_fp11** versions of 8xSIMD accelerators has no support for pipelined vector floating point division and it is suitable for applications like FIR filters or adaptive LMS filters with no need for floating point division.

The **bce_fp12** versions of 8xSIMD accelerators are larger in comparison to the bce_fp11 equivalents and support in a single data path the pipelined vector floating point division. Accelerators are suitable for applications like adaptive normalised NLMS filters and the square root free versions of adaptive RLS QR filters and adaptive RLS LATTICE filters.

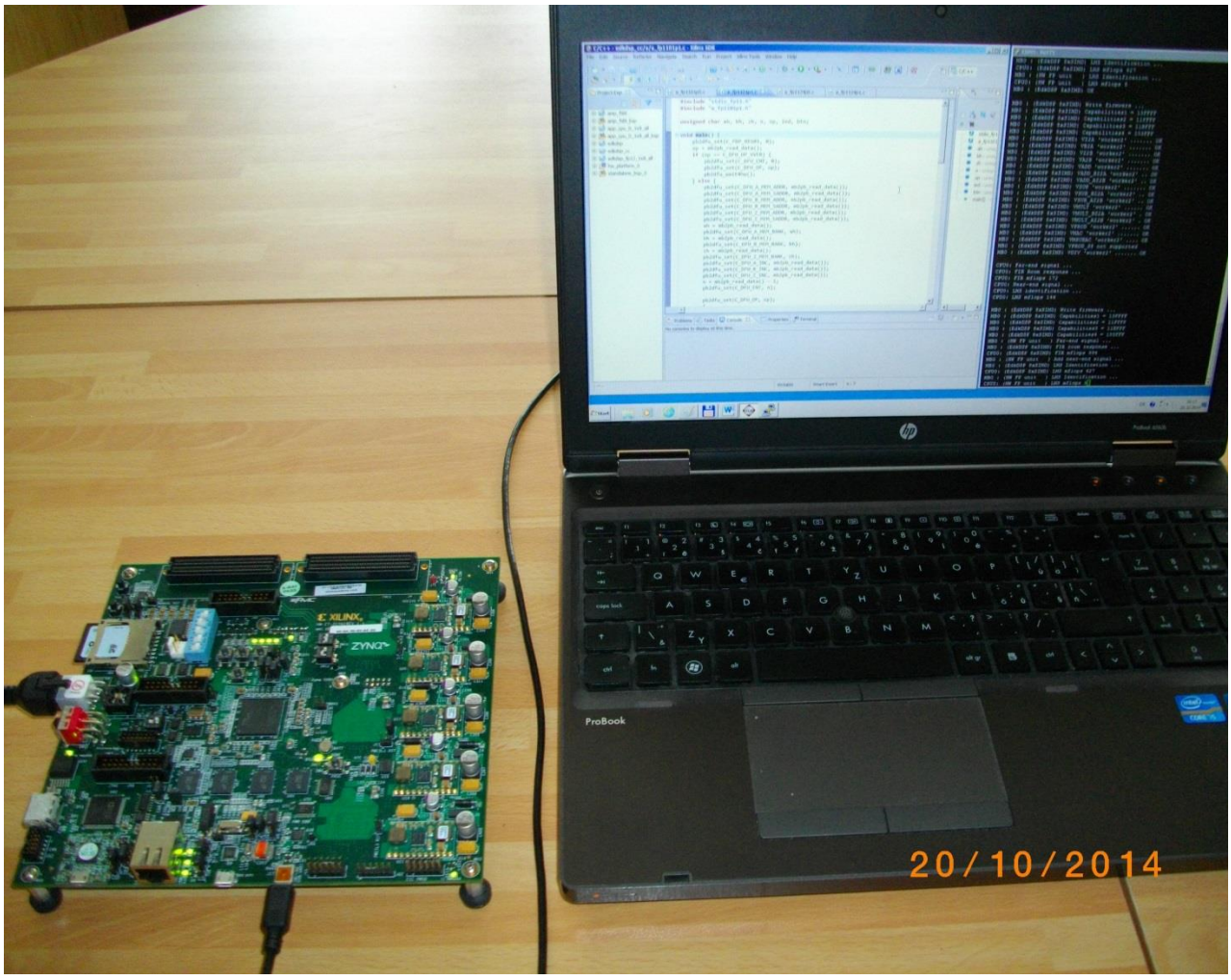

Figure 3: Asymmetric multiprocessing on ZYNQ with ARM, Microblaze and 4x(8xSIMD) EdkDSP floating point accelerators. Vivado 2013.4 evaluation design is running on Xilinx ZC702 board.

2.2 Reconfiguration of the entire Zynq FPGA programmable logic (PL)

Typical application where the *partial dynamic reconfiguration* of the FPGA is used divides the programmable logic area into "static", which doesn't change and runs without interruption and the "dynamic part". In the most cases, the static part of FPGA contains microprocessor and the dynamic part implements typically application dependent acceleration or IO interfaces.

Analogically to the typical reconfigurable design described above, the Zynq device in the AMP platform was partitioned to the dual ARM A9 CPU cores in static part (they are implemented on Zynq die) and to the dynamic part where the entire Zynq PL (FPGA part of Zynq) is used. This arrangement is much simpler than implementation of the partial dynamic reconfiguration (in terms of design development and testing). It is cheaper as well, since the partial dynamic reconfiguration license is not required. The main drawback of the approach is the higher reconfiguration time.

The summary of pros and cons for both kinds of reconfiguration available in AMP platform on Zynq can be found in Table 1.

Reconfiguration type	Pros	Cons
FW reconfiguration	- Ultra fast (1CC for firmware memory switch) - ASIC implementation possible - Complex DSP operations can be broken into individual steps, each implemented as one firmware - Isolation of firmware developer from development of hardware	- functionality is limited by margins given by DSP unit implementation
PL reconfiguration	- Slow (tens of milliseconds, but can be mitigated by employing partial bit streams to hundreds of microseconds) - No need for partial reconfiguration license	- Simple implementation compared to partial reconfiguration - Faster and cheaper development - Can change whole PL design

Table 1: Summary of pros and cons of the reconfiguration types in AMP platform on Zynq

2.2.1 API Support

In general, each API used by the user application for the dynamic reconfiguration must share the same architecture as shown in Figure 4. The API must implement the following operations related directly to reconfiguration support:

Operation	Description
`init`	Initialization of reconfiguration subsystem
`stop`	Stops the operation of the reconfigurable component cooperatively (after end of operation cycle) and disable its IO interfaces
`kill`	Stops the reconfigurable part immediately and disable its IO
`reconfigure`	Loads new configuration into reconfigurable component
`start`	Enables IO and starts execution of reconfigurable section

Table 2: Basic functions needed for the dynamic reconfiguration

In case of AMP platform on Zynq processor, the reconfiguration of EdkDSP accelerator is performed by writing to its firmware memory. The speed of the memory interface is then given by the clock frequency on the memory port, i.e. CLK_FREQ*4 MB/s (400MB/s at 100MHz clock for example). Similarly, the reconfiguration of the PL is executed using DMA & PCAP interface of the Zynq. Its speed is given in the non-encrypted case as 400MB/s (100 MHz clock).

API operation	Implementation	
	Firmware reconfiguration	PL reconfiguration
init		PcapInit()
stop		PLstop();
kill	wal_reset_worker()	PLreset();
reconfiguration	wal_set_firmware()	PcapLoadPartition()
start	wal_start_operation()	PLstart();

Table 3: Basic functions implemented by UTIA for the dynamic reconfiguration on Zynq

The Zynq platform with asymmetric multiprocessing has been documented in the electronically published application notes [16] and [17].

The corresponding evaluation packages with the precompiled designs can be downloaded from [18] and [19]. UTIA is already supporting the mature Xilinx ISE 14.5 design flow in [16], [18] and the new state of the art Xilinx Vivado 2013.4 design flow [17], [19]. Firmware based dynamic reconfiguration of floating point accelerators is already implemented in these two evaluation packages and it is already evaluated.

The dynamic reconfiguration of the Zynq programmable logic part will be implemented, documented and tested in the second year of the EMC^2 project.

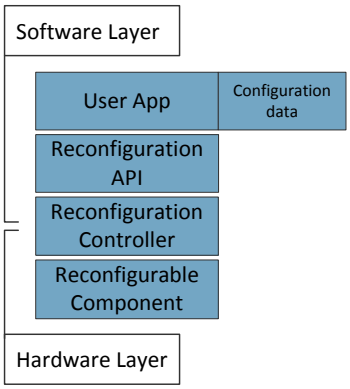

Figure 4: Reconfiguration controller and API

3. Reliable and Self-Healing Dynamic Reconfiguration Manager (DRM) on LADAP platform (Virtex 5)

Dynamic Partial Reconfiguration (DPR) for FPGA devices has many advantages not only for the Space domain but also for many other industrial domains. Nevertheless, the space domain is a very demanding environment exposed to high radiation doses and where reliability of systems is critical. Taking into account these issues, in the context of subtasks T4.3 and T4.4 and the living lab 3 (Space applications) a platform will be built which allows dynamic reconfiguration of a Xilinx FPGA but including reliability and self-healing features. This platform is developed in close cooperation between Tecnalia and Thales Alenia Space (TASE); Tecnalia is in charge of the development of the platform's programmable logic whilst TASE provides the hardware board (LADAP) and feedback about requirements of the space domain.

The proposed platform will have the following characteristics:

- Designed for Virtex 5Q Devices (certified for Space environment) and implemented on the LADAP hardware board (description to be found on oncoming D9.2 [3]) developed by TASE
- The core of the platform will be a Microblaze processor which controls both the reconfiguration process and some of the fault-tolerant features.
- It allows Dynamic Reconfiguration of custom peripherals in the Virtex 5 FPGA device
- It includes fault-tolerant features against radiation induced errors both for error mitigation (Xilinx Isolation Design Flow methodology, watchdog timer interrupts …) and error detection and correction (periodic configuration memory scrubbing, continuous check of configuration memory, read-back CRC of configuration memory …)
- The platform can be controlled by the software implemented in the LEON processor of the LADAP platform through a communication interface implemented as a shared dual port RAM memory.

3.1 Module Description

3.1.1 Architecture

The architecture of the DRM is shown in the following figure:

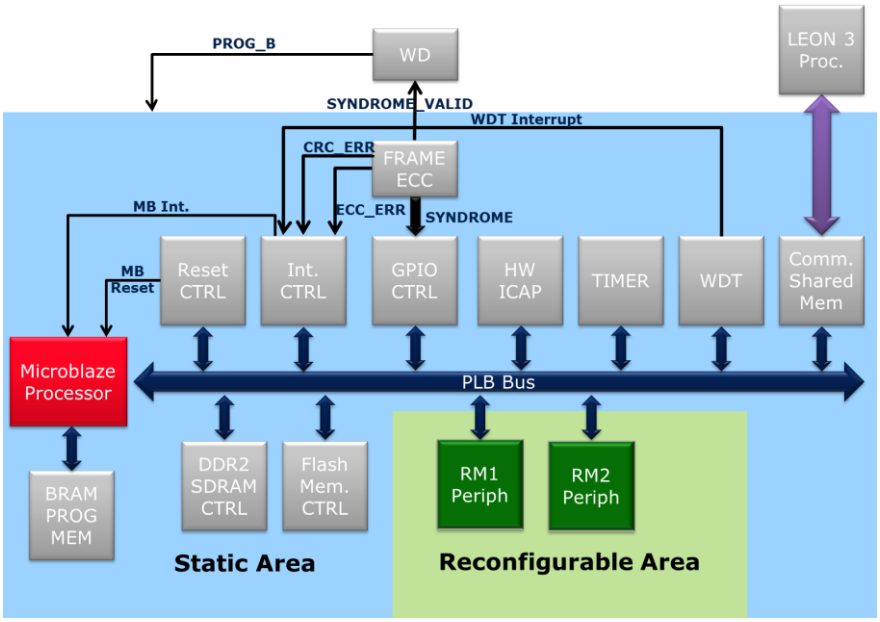

Figure 5: DRM architecture

As it has been already mentioned the FPGA device is divided in 2 different partitions: A reconfigurable area when the different modules which are potentially reconfigurable are implemented and a static one where the DRM is implemented. The design in the static area has a higher criticality level as it is in charge of controlling the rest of the modules implemented in the FPGA and also of the communications with the control software in the LEON3 processor; therefore more error protection mechanisms should be implemented on it. The DRM has the following modules:

- **Microblaze processor**: Embedded processor implemented using logic resources. It's the core of the system in charge of controlling the dynamic reconfiguration of peripherals, some of the error protection functions and configuration of the system. The fault-tolerant feature of the processor will be activated. The processor accesses its peripherals with a PLB bus.

- **Program memory**: It is the 64 Kbits code and data memory used by the Microblaze processor. It is implemented using Block RAM memory resources of the device. These resources are protected with the ECC feature to detect and correct radiation induced errors.

- **Reset and Interrupt controllers:** Peripherals in charge of controlling the reset and interrupt request (IRQ) inputs of the Microblaze processor. They receive the different reset and interrupt sources and generate the appropriate reset and IRQ signals for the processor.

- **FRAME_ECC primitive:** It's a dedicated resource of Virtex 5 devices in charge of autonomously monitoring the configuration memory. It generates three different output signal:
    - A SYNDROME_VALID signal each time it reads a new configuration frame; this signal is used as a "heart beat" by an external watchdog for SEFI detection.

  - An ECC_ERROR signal together with a SYNDROME signal each time it detects an erroneous frame. The ECC_ERROR is the error flag and the SYNDROME shows the address of the faulty frame so the processor can repair it via the HW ICAP.
  - A CRC_ERROR flag that shows that there is an error in the configuration memory that cannot be located and therefore full memory scrubbing is required.

- **HW ICAP controller:** It is the peripheral in charge of controlling the internal configuration port of the device used during the dynamic reconfiguration process. Information from the partial configuration images is loaded into the FPGA configuration memory through this peripheral.

- **Watchdog Timer:** It is a safety mechanism to avoid processor's crashes. It is a continuously running timer that generates an interrupt signal for the processor each time it overflows, the processor then resets the timer. If the counter overflows twice without a processor reset, then it generates a reset signal that restarts the whole system.

- **Communication Interface Shared Memory:** It is a dual port shared memory that is employed as a communication interface with the control software running on the LEON3 processor of the LADAP board. It is implemented using block RAM memory resources of the FPGA and it is divided in two different areas: In the first one the LEON3 processor writes messages and the Microblaze reads them and the other one vice versa.

- **External Memories Controllers:** The system has access to two different external memories: a SDRAM memory and a flash one. The flash memory is employed to store the full and partial configuration images files and the SDRAM memory could be used for storing program data.

- **Timer**: A timer peripheral that could be employed for timing functions of the software running in the Microblaze.

Apart from the DRM implemented in the static area of the device, additional modules for different applications can be implemented in the reconfigurable area. These modules are accessible to the Microblaze also through the PLB bus, so it can monitor and configure their working. As mentioned before, their functionality can be dynamically changed by the DRM. These modules have a lower criticality level and, therefore, a lower number of error protection mechanisms are implemented on them. As proof of concept, two different types of modules are implemented: one for a simple mathematical operation (addition, subtraction or multiplication) and another one implementing different FIR filters.

3.1.2 Error Protection Techniques

As it has been already mentioned, a set of different error protection techniques is implemented on the DRM to provide fault-tolerant features. These techniques can be classified in two general types: those techniques oriented to mitigate or restrict the effect of potential errors and those ones which enable it to detect and correct them. Regarding the first type, two techniques are considered:

Mitigation technique #1: Xilinx Isolation Design Flow
Xilinx Isolation Design Flow (IDF) is a design technique defined by the FPGA manufacturer Xilinx [4], which allows for fault-containment at a FPGA module level. It makes use of the standard Xilinx design tools and requires attention to the design layout by respecting strict set of rules. The objective of this technique is to isolate the impact of single failures to a specific module reducing drastically their effect on the system.

IDF consists of first partitioning the design in different hierarchical modules and afterwards add the design layout and routing constraints so that each partition is restricted to a closed area of the device, isolated from the rest of the design by a fence of unused logic resources.

Mitigation technique #2: WatchDog timer for Microblaze processor
As it has been mentioned in the previous chapter, a watchdog timer peripheral has been included in the DRM in order to restart the processor if it crashes.

Additionally, three protection mechanisms are proposed to detect and correct errors:

Detection and Correction technique #1: memory scrubbing or configuration management
The focus of this technique is the prevention of SEU accumulation in the FPGA configuration memory. Since our goal is to design and develop a self-managed FPGA and having into account that the Virtex-5QV FPGA has an extremely low upset rate in configuration bits of 5 events per year (see reference [5]), an internal memory scrubber scheme will be implemented.

Configuration memory scrubbing consists in re-writing the FPGA configuration memory while the design is actively running. This is different from a full device reconfiguration, which implies deleting the configuration memory before reloading the full bit stream. Configuration memory scrubbing is based on active partial reconfiguration of the desired configuration frames.

Since the internal configuration management solution makes use of the internal ICAP port, and since the design does include an embedded processor core accessing the ICAP port to control the reconfiguration process of the respective partitions, the configuration memory scrubber will be implemented using the embedded processor instead of an external entity.

Mitigation technique #2: SEFI detection and mitigation
SEFI detection will be implemented from two sources: the FPGA configuration registers as shown in XAPP588 [6] and externally via a watchdog device (following XAPP1090 [7]). This watchdog will receive the SYNDROME_VALID signal from the FRAME_ECC as a heart-beat reset. If the signal is not received (which indicates a SEFI) the WD overflows and generates a flag signal.

Mitigation or rather recovery actions, in all cases, require to pulse the PROGRAM_B pin which means a full device reconfiguration. Having to fully reconfigure the FPGA is not worrying, since the probability of device SEFIs is really very low [8]

Mitigation technique #3: Microblaze softcore protection
We will enable the Microblaze fault tolerance features, which are described in Xilinx user guide UG081 (see [9]).
These fault tolerance features can be summarized as follows:

- Memory protection:
  - LMB Block RAMs protected with ECC
  - Block RAM in the instruction and data caches protected with parity bit
  - Block RAM in the MMU UTLB (Memory Management Unit Unified Translation Look-Aside Buffer) protected with parity bit
  - Block RAM in the branch target cache protected with parity bit
- LMB BRAM Controller with different modes (Minimal, Small, typical and full). The typical LMB BRAM Controller use case will be implemented.

3.2 Development timeline

The development of the DMR will consist of the following phases:

- Phase I: Architecture and interface Specifications: **Finished on December 2014**.
- Phase II: Design and Development of the basic DRM system (**from December 2014 to August 2015**) including:
  - Specifications of the system's design.
  - Test Plan definition
  - Design and development.
  - Validation and Tests.
- Phase III: Design and Development of the error protection techniques (**from August 2015 to May 2016**) including:
  - Specifications of the mechanism's design.
  - Test Plan definition
  - Design and development.
  - Validation and Tests.
- Phase IV: Fault-tolerant reconfigurable peripherals design (**from May 2016 to December 2016**) including:
  - Specifications of the mechanism's design.
  - Test Plan definition
  - Design and development.
  - Validation and Tests.

4. Reliability Analysis in Presence of Hardware-induced Errors

To assess the required level of redundancy, the underlying system has to be analysed with respect to its reliability. To do so in a systematic fashion, we have developed a classification framework to analyse and model the physically induced reliability violations [10]. Determining the amount of required redundancy as well as to specify which type of errors are visible on software level, a connection has to be built between the occurrence of hardware errors and how they propagate through the system.

A lot of research has addressed the mitigation of errors. This has resulted in a variety of methods with varying degree of error correction capabilities and costs in terms of computational effort and/or power. Over the years, it became clear that cross layer approaches are most promising to achieve efficient trade-offs. Such cross layer approaches combine monitoring, control and correction mechanisms on multiple layers in the design.

In recent years a new trend has been started. It takes into the account, that the input data to the system are not free of errors.

Examples of these cases are noise in captured images. As an example, median filters are used to cover faulty pixels in an image sensor. The human perception can accept slight increases in noise of an image without notice. This can be used in the image processing domain as base for the graceful degradation in cases, where power or computing resources are not sufficient. Moderate error rates in the output (the image or video stream) are still considered to be acceptable.

Another domain is the processing of wireless or wireline data streams, where input data is subject to major distortions. The processing in the transmitting and receiving paths assure that the finally decoded data is still within certain error bounds. This combines two aspects that are typically not assumed in digital processing systems:

- the systems behaves well despite errors in the data, and
- data is not 100% reliable, in fact there is an accepted and well defined upper limit on the error probability.

In recent years, these aspects have been taken into considerations, when designing even the digital part of such communication channels [11], [12], [13].

We have shown successfully that for memories such a trade-off between rare error events and HW/SW to mitigate such errors can lead to dramatic reduction in energy per operation (cf. Figure 6, [14]). To expand this approach into the domain of the digital processing, we are developing a framework that allows an efficient assessment of the impact of hardware errors on the system behavior [15]. Such a framework has to go beyond simple signal-to-noise-ratio (SNR) metrics as it is not capable to capture the impact of errors on a system level. Also, the correlation of errors has to be quantified properly. The simplified assumptions of uncorrelated errors are often not valid. In our contribution to the INDIN, we show the impact of such simplifications and sketch a computational efficient framework to realize the propagation of errors.

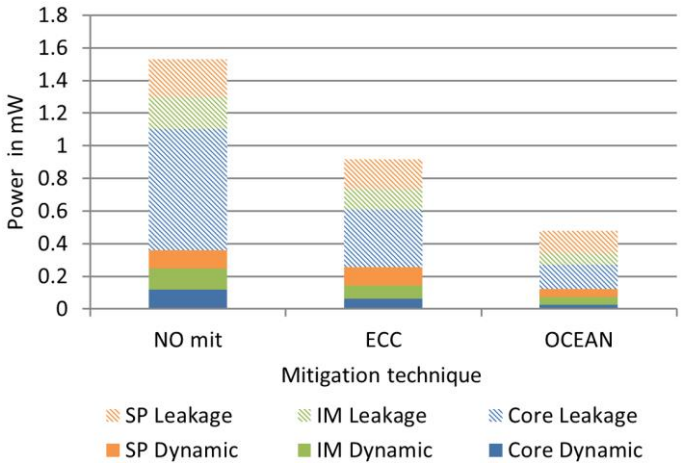

Figure 6: Power consumption applying different error mitigation schemes [14]

5. Resilient Reconfigurable Multiprocessor arrays: probabilistic analysis of availability and efficiency

5.1 Concrete work and time plan in T4.4

In T4.4, Chalmers will perform a probabilistic analysis to evaluate the availability of the reconfigurable multicore architecture designed in T4.2. In T4.2 Chalmers designs the microarchitecture of a reconfigurable processor array which is able to provide coarse-grain and fine-grain reconfigurability to mitigate permanent faults as depicted in Figure 7. Coarse grain reconfigurability will allow replacement of faulty processor parts by identical (presumably spare parts) borrowed by a neighboring processor. To exemplify, a pipeline stage of a processor in Figure 7 (IF, DC, EX, or MEM) could be replaced when damaged by the same, identical stage of the neighboring processor using the switches and vertical wires tha connect the two processors. Fine-grain reconfigurability will offer further resources to instantiate more processor parts to replace faulty processor components, acting as a wild card.

Each processor is partitioned in decoupled pipeline stages (vertical partitioning), while the execution stage is further split in three concurrent parts (horizontal partitioning) due to its size. When a part of EX is implemented in the fine-grain logic, the entire stage is further pipelined in two stages to improve speed.

In the first year, the above mentioned concepts of the proposed reconfigurable multicore architecture will be formalized to allow to analytically quantifying their efficiency.

In the second period of the project, Chalmers will describe a probabilistic analysis for calculating the availability of the above mentioned processor array. That is an analysis, which calculates the probability of correcting a given set of faults. We will assume a particular distribution of faults based on which the probability of correction will be retrieved as well as the average number of working processors for a given number of permanent faults. In addition, we will estimate the probability of failures (to correct) for a given number of processors expected to function in the system before failure.

In the last period of the project, we will further analyze the average distance of substituted blocks and accordingly estimate performance and energy efficiency of the reconfigurable processors for various fault densities.

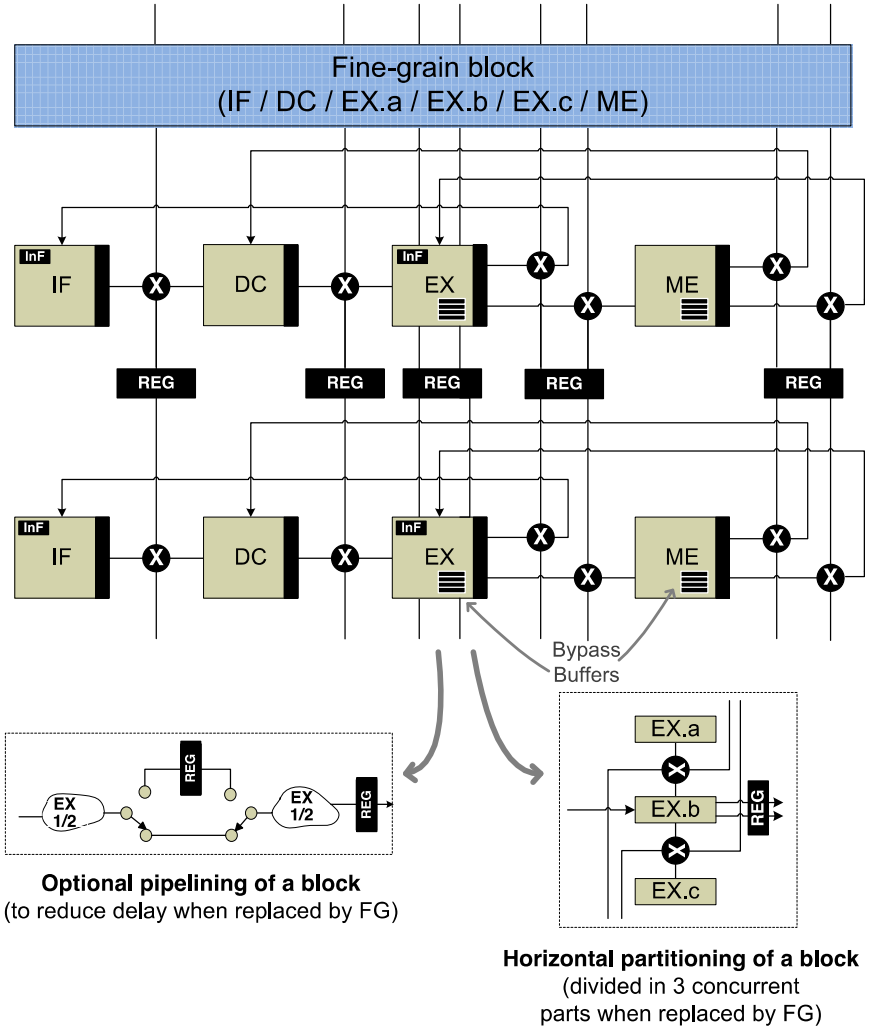

Figure 7: The proposed reconfigurable resilient multiprocessors using the coarse and fine-grain reconfigurable substrate. Coarse-grain reconfigurability allows a (damaged) pipeline stage of a processor to be replaced by the identical stage of the neighboring processor (i.e. the DC stage of the upper processor to be preplaced by the DC stage of the lower processor). Fine-grain reconfigurability allows a (damaged) pipeline stage to be replaced by a new instance of it configured in the fine-grain block.

5.2 Expected outcome/highlights of the contribution

The work described above will allow us to quantify the availability of the proposed adaptive reconfigurable multicore architecture in the first place. That is the number of faults sustained in a system composed of the proposed processors before failure, as well as the average availability of the multicore array for different fault densities. We aim for an analytical model for calculating the above, which will thereby avoid the computationally intensive exhaustive fault injection, which is in most cases too slow. Moreover, this work will reveal the efficiency of the proposed reconfigurable multicore array. More precisely our work will allow evaluating the performance and energy efficiency of our architecture for different fault densities (which require different reconfigurations).

As a final step of our study, this work will enable us to perform a design space exploration of our approach and conclude on the most efficient granularities (size of substitutable components on a reconfigurable multicore array) as well as the most efficient mix of fine and coarse-grain reconfigurability that maximizes the availability, performance and energy efficiency of the design for an expected fault density.

6. Conclusions

The deliverable *D4.6 – Specification of preliminary architecture and API respecting the requirements for error handling and redundancy of accelerators* describes preliminary solutions for the dynamically reconfigurable subsystems which can eventually be used in living labs/demonstrators. It describes the planned reconfigurable architecture structures considered for the development within the EMC2 WP4.

All 4 proposed contributions outline concrete approaches to dynamic reconfiguration resulting serving to the increased error handling capabilities and the increased redundancy of accelerators.

UTIA contribution is addressing 2 forms of reconfiguration of HW as part of the asymmetric multiprocessing on the 28nm ZYNQ device. Floating point accelerators are reconfigured in the run-time by change of the firmware of internal scheduling controllers. The second level of dynamic reconfiguration is relying on the ZYNQ dual-core ARM processor. The cost aspects and the availability of the tool chain for the industrial are also respected by restriction to the dynamic reconfiguration of the complete programmable logic fabric of the ZYNQ device.

Tecnalia work will improve error-handling capabilities and the increased redundancy by implementation of the complete Dynamic Partial Reconfiguration (DPR) for FPGA for the space domain. This work is performed in relation to the living lab 3 (Space applications).

IMEC-NL is addressing the required level of redundancy with respect to the reliability of the system in a systematic fashion. The investigation will focus on the relations between the occurrence of hardware errors and error propagation through the system. IMEC is proposing to use the cross layer approaches to achieve efficient trade-offs. These cross layers will combine monitoring, control and correction mechanisms on multiple layers in the design.

Finally, Chalmers is proposing a microarchitecture that is formed by a reconfigurable processor array which is able to provide coarse-grain and fine-grain reconfigurability to mitigate permanent faults. Coarse grain reconfigurability will allow replacement of faulty processor parts borrowed by a neighboring processor. Fine-grain reconfigurability will offer further resources to install more processor parts to replace faulty processor components.

The specified preliminary architecture will contribute and impact these results of the EMC2 project.

- The dynamic reconfiguration of the programmable logic on the ZYNQ asymmetric multiprocessing processing platform (Leading partner - UTIA).
- Partial dynamic configuration on Xilinx Virtex 5 FPGA for the use in Space application (Leading partner - Tecnalia).
- SW/HW cross layer solution will be developed. It will be combine the monitoring, control and correction mechanisms on multiple layers in the design. (Leading partner - Imec-NL).
- Specific microarchitecture will be developed. It will be formed by a reconfigurable processor array which is able to provide coarse-grain and fine-grain reconfigurability to mitigate permanent faults. (Leading partner - Chalmers).

The deliverable D4.6 provides the basic specification for the development and implementation work related to the error handling and redundancy of accelerators within the EMC2 WP4 work package. It relates mainly to the Task 4.4 and also to several demonstrators of WP4 and living labs (mainly Space).

7. References

[1] http://www.xilinx.com/support/documentation/application_notes/xapp1093-amp-bare-metal-microblaze.pdf
[2] http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/boards_and_kits/zynq-7000_soc_boards_and_kits/zynq-7000_soc_zc702_evaluation_kit.html
[3] Deliverable 9.2 – Space Application Concept Report: Expected for May 2015
[4] Xilinx. WP412, The Xilinx Isolation Design Flow for Fault-Tolerant Systems, v1.1, October 2013.
[5] Xilinx. XAPP1073, NSEU Mitigation in Avionics Applications, v1.0, May 2010.
[6] Xilinx. XAPP588, Virtex-5QV FPGA External Configuration Management, v1.0, January 2012.
[7] Xilinx XAPP1090, Virtex-5QV FPGA Internal Configuration Management Implementation
[8] Gary Swift and Gregory Allen. VIRTEX-5QV Static SEU Characterization Summary. Xilinx Inc. and Jet Propulsion Laboratory, Second Release, May 2013.
[9] Xilinx. XAPP887, PRC/EPRC: Data Integrity and Security Controller for Partial Reconfiguration, v1.1, June 2012.
[10] D. Rodopoulos, "Classification Framework for Analysis & Modeling of Physically Induced Reliability Violations," ACM Computing Surveys, Vol. 47, Issue 3, Feb. 2015.
[11] A. Khajeh et al., "Error-Aware Algorithm/Architecture Coexploration for Video Over Wireless Applications," Trans. Embed. Comput. Syst., 2012.
[12] C. Gimmler-Dumont et al., "A Cross-Layer Reliability Design Methodology for Efficient, Dependable Wireless Receivers," Trans. Embed. Comput. Syst., 2014.
[13] G. Karakonstantis et al., "On the exploitation of the inherent error resilience of wireless systems under unreliable silicon," Design Automation Conference (DAC), 2012.
[14] T. Gemmeke, J. Stuijt, M.M.S. Aly, P. Raghavan, D. Atienza Alonso, F. Catthoor, "Resolving the Memory Bottleneck for Single Supply Near-Threshold Computing," Design Automation and Test in Europe (DATE), 2014.
[15] G. Psychou, T.G. Noll, T. Gemmeke, "On the use of Analytical Techniques for Reliability Analysis in Presence of Hardware-induced Errors," submitted to INDIN, 2015.
[16] http://sp.utia.cz/results/Utia_EdkDSP_145_ZC702/Utia_EdkDSP_145_EMC2_ZC702.pdf
[17] http://sp.utia.cz/results/Utia_EdkDSP_Vivado_2013_4_EMC2/Utia_EdkDSP_Vivado_2013_4_EMC2_ZC702.pdf
[18] http://sp.utia.cz/index.php?ids=results&id=Utia_EdkDSP_145_ZC702
[19] http://sp.utia.cz/index.php?ids=results&id=Utia_EdkDSP_Vivado_2013_4_EMC2