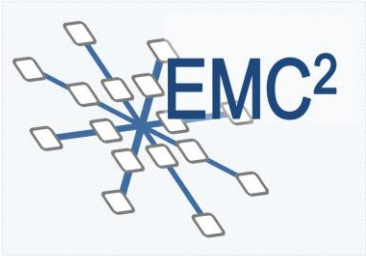


Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	**D4.17 – Final demonstration platforms including basic innovative techniques**	
Work package	WP4	Multi-core hardware architectures and concepts
Task / Use Case		
Subtasks involved	T4.2, T4.3, T4.4, T4.6	
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto:werner.weber@infineon.com	
Deliverable responsible	IFAT Alexander Lipautz, alexander.lipautz@infineon.com + 43 (5) 1777 6595	
Version number	v1.0	
Date	03/10/2016	
Status	Final	
Dissemination level	PU	

Authors

Partici-pant no.	Part. short name	Author name	Chapter(s)
04D	UTIA	Jiri Kadlec	Chapter 1 Introduction
01V	TUBS RUBS TUDO	Rolf Meyer, Jan Wagner, Lilian Tadros	Chapter 2 EMC² SoCRocket – EMC² Virtual Platform Framework
04D	UTIA	Jiri Kadlec, Zdenek Pohl, Lukas Kohout	Chapter 3 Asymmetric Multiprocessing on ZYNQ and HDMI I/O demonstrator
15O	SevenS	Javier Diaz, Benoit Rat, José Luis Gutiérrez	Chapter 4 Dependable and high accuracy time transfer for distributed control applications
02A 01L 01A 02 F	AVL/GRZ EB IFAG VIF	Eric Armengaud, Peter Priller Alexander Mattausch Werner Weber Allan Tengg	Chapter 5 Automotive domain: Network technologies and connection to testbed (based on UC7.3)
02D 02E	TTT TUW	Andreas Eckel Haris Isakovic	Chapter 6 Multi-processor System-on-Chip Architecture for Cross-domain applications
02C	IFAT	Thomas Herndl, Norbert Druml	Chapter 7 Automotive Time-of-Flight Demonstrator
18M	ENSILICA	David Wheeler	Chapter 8 Fine Grain Partially Reconfigurable Array of Processors on Zynq with Linux reprogramming support
18D	Sundance	Emilie Wheatley	Chapter 9 HDMI-in video processing to HDMI-out controlled by Ethernet port
11K	POLITO	Massimo Violante	Chapter 10 Demonstrator platform for multicore avionic systems
18H	UoBR	Steve Kerrison	Chapter 11 MCENoC Demonstrator: A Network-on-Chip for Mixed-Criticality Embedded Systems (18H UoBR)
02C	IFAT	Alexander Lipautz	Chapter 12: Conclusions

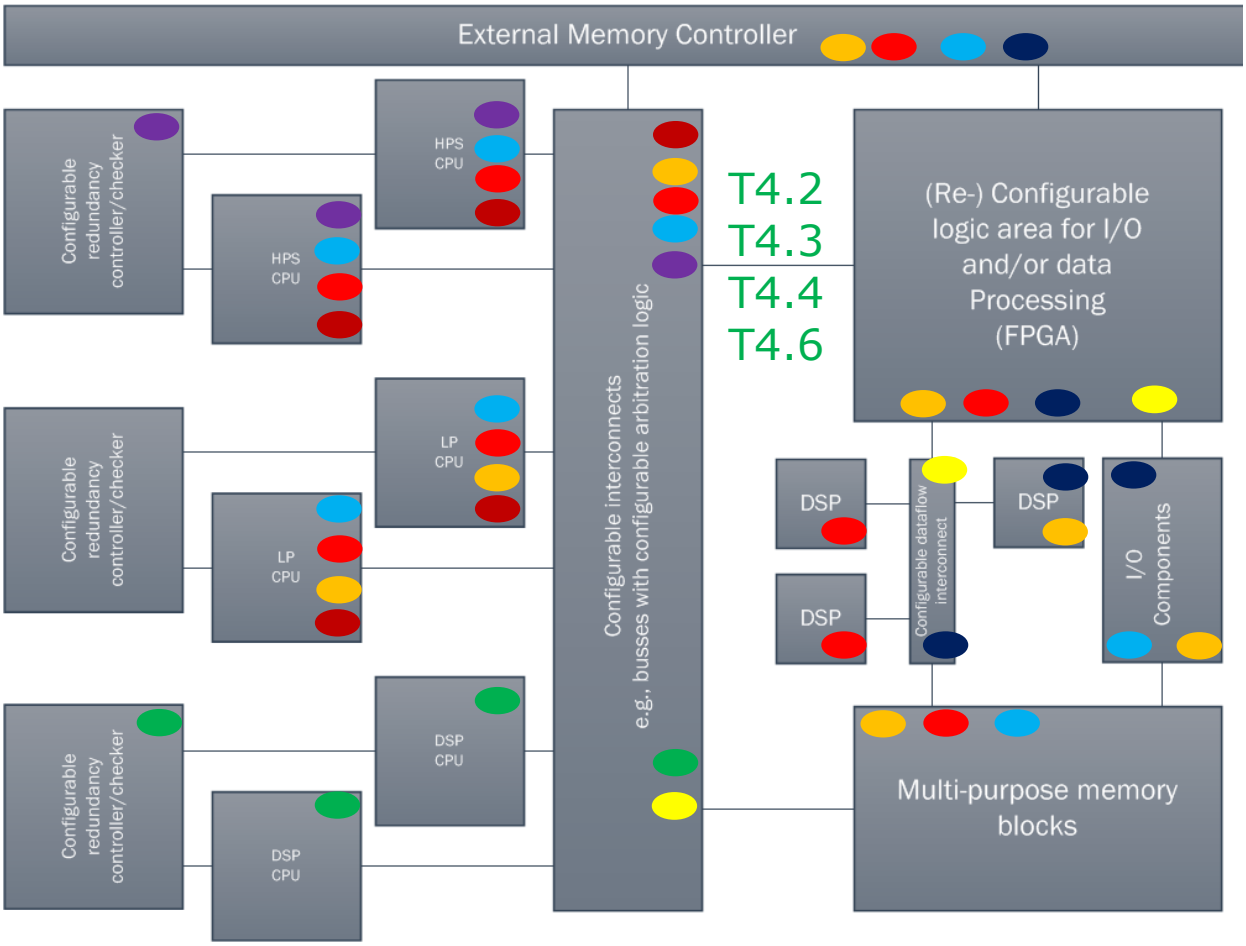
Document History

Version	Date	Author name	Reason
v0.1	11/07/2016	Alexander Lipautz (IFAT)	Initial template
v0.5	19/09/2016	Alexander Lipautz (IFAT)	Ready for review
v1.0	03/10/2016	Alexander Lipautz (IFAT)	Final version
	03/10/2016	Alfred Hoess	Final editing and submission

Publishable Executive Summary

This deliverable is presenting enhanced demonstration platforms including basic innovative techniques in form of demonstrators prepared by WP4 partners in the EMC² project. It describes following demonstrators:

- EMC² SoCRocket - transaction-level modelling framework for space applications
- Asymmetric Multiprocessing on ZYNQ and HDMI I/O demonstrator
- Dependable and high accuracy time transfer for distributed control applications
- Automotive domain: network technologies and connection to testbed (based on UC7.3)
- Multi-processor system-on-chip architecture for cross-domain applications
- Automotive time-of-flight demonstrator
- Fine grain partially reconfigurable array of processors on Zynq with Linux reprogramming support
- DVI-in video processing to DVI-out controlled by Ethernet port
- Demonstrator platform for multicore avionic systems



WP4 tasks T4.2 T4.3 and T4.4 cooperating with T4.6 in the D4.16 demonstrators development

These enhanced demonstration platforms including basic innovative techniques milestone present the status of developments within the WP4 package in the middle of the EMC² project. Demonstrators form concrete base for the collaboration and developments in the EMC² project.

Table of contents

List of figures

1. Introduction

WP4 of the EMC^2 project deals with new multi-core hardware architectures and concepts.

This deliverable is presenting demonstrators provided by WP4 partners of the EMC^2 project. Each contribution is describing an overview of the demonstrator and it is pointing to detailed information and documentation provided by individual partners.

We have selected this format to motivate the bottom up cooperation of partners within WP4, with other technical work packages, and cooperation together with the Living Labs partners of the EMC^2 project.

Workpackage 4 is focusing on four technology lanes:

(1) Heterogeneous Multiprocessor SoC architectures
(2) Dynamic reconfiguration on HW accelerators and reconfigurable logic
(3) Networking
(4) Virtualization and verification technologies

This deliverable describes WP4 demonstrators for Technology Lanes (1), (2), (3) and (4).
Contributions to these technology lines are indicated in each chapter. The concrete technical topics are also visualized on the symbolic „architecture“ by a bullet.

2. EMC² SoCRocket – EMC² Virtual Platform Framework

Increasingly large portions of electronic systems are being implemented in software and its development cost starts dominating the overall system's cost. Software is also becoming the critical part of the development schedule, mainly because deploying and testing it on the real target hardware is complicated.

Transaction Level Models (TLMs) can be used to describe both, timing and functionality, of system components and their communication interfaces at a high abstraction level. Embedded in a virtual platform, these models are sufficiently accurate to not only allow early software development and verification in a realistic environment but also functional verification of the modeled hardware. The capability of early design-space exploration is therefore a vital building block of full hardware/software co-design.

To achieve these goals, we designed the *SoCRocket Framework*. Written in *SystemC/TLM*, it is fitted to serve the industry's special needs. It already builds the foundation of space-domain ESL design. We tied together the following features to enable the construction of virtual platforms for various applications:

- **Models** - All models are designed to simulate their corresponding counterparts from the Aeroflex Gaisler GRLib (also available under GPL)
- **Platform Generator** - Easy configuration via GUI or from the command line
- **Automation Tools** - To run big batches of design-space explorations
- **Infrastructure** - Reusable components for building new components at ease
- **Build System** - Extended build system for compiling models, platforms, target software, RTL co-simulations, and regression tests

2.1 Overview

Our activities in this project aim towards implementation and validation of an efficient methodology allowing software to be developed before the final hardware is ready. We have therefore developed a design flow for the implementation of virtual platforms (VP) based on TLM in SystemC.

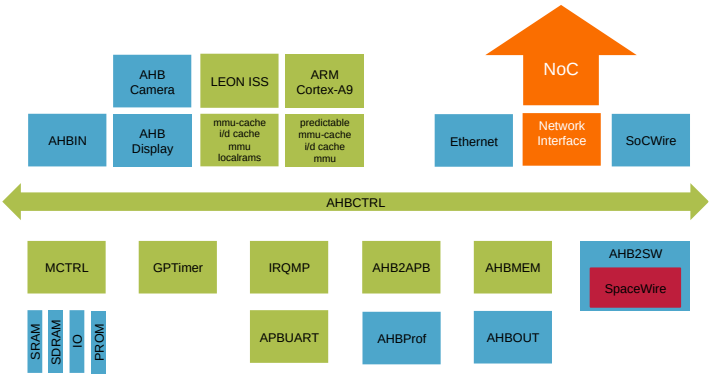


Figure 1: SoCRocket Models

To support the developed design flow it was mandatory to stock it up with basic models and a supportive foundation. This allows the users to be faster through reuse of already existing components. It also gives model developers a starting point to derive their models from.

The outcome of this project is now bundled as a library, which we call the SoCRocket – Virtual Platform Framework.

Hence the free availability of the Aeroflex Gaisler GRLib[1] Components we designed our Models to behave alike, therefore all models found in the SoCRocket Framework hava a corresponding RTL component in the GRLib. Platforms constructed and tested in our Framework can easily be migrated to the GRLib and simulated and synthesized at RTL level. Furthermore, the developed models go beyond the GRLib with the integration of the ARM CortexA9 and MicroBlaze Processors. This enables the project partners to simulate systems closer to the UTIA+Sundance platform (Chapter 3). For more information, see WP2/WP4 Quadcopter cooperation.

2.2 Infrastructure

As foundation (orange) for our library we selected a series of appropriate infrastructure components and built a layer of abstraction upon them (red).

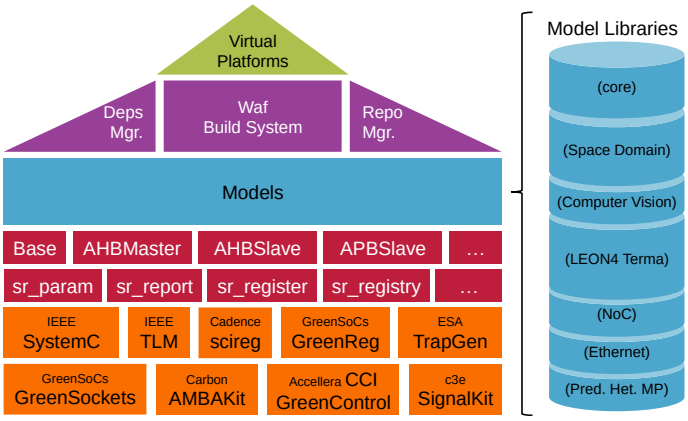


Figure 2: SoCRocket Structure

We contribute a modeling kit for fast TL signal communication to the foundation. Throughout the library the SignalKit is used to model interrupt lines, reset and snooping. As well as an extended logging mechanisms, sr_report, especially useful for simulation observation and output. It can be configured in seven levels that can be set globally or for a single module. A model registry (sr_registry) can be used to dynamically instantiate models and prevent recompilation. And last but not least we extended the gs_params with meta parameters to allow easy handling of meta information in addition to generics, registers and other non-memory mapped data.

[1] http://www.gaisler.com/index.php/downloads/leongrlib

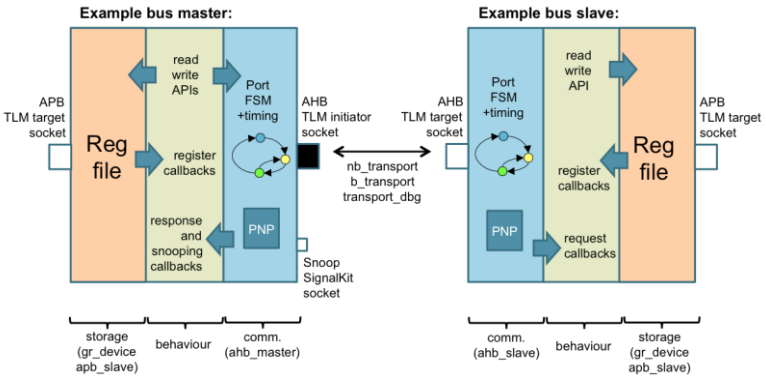


Figure 3: Model design

One of the main purposes of the library is to leverage the development of new components and to test them on system level before going to RTL. To make this as easy as possible we provide a set of library base classes encapsulating the communication and the storage part of the models. All our models, except the AHBCTRL are structured this way.

On the left side of the previous figure you see a typical AHB master component. The model is a C++ class which inherits a register file and an APB slave socket from class APBSlave and an AHB master socket from class AHBMaster. The user ought to fill out the behavior, which is the green part in the middle. For access to the sockets there is a simple read/write API. The register file can be equipped with callbacks for read and write operations to any register or bit field. In AT mode the port interface triggers a response function in the behavior, as soon data is available at the socket.

Slaves are structured in similar ways. Only this time the implementing module inherits from class AHBSlave giving it an AHB TLM target socket. For any request the slave interface triggers a callback in the behavior. All details of the state machines in the front-end are hidden.

2.3 Conditions for use, downloads

In the last 30 months TU Braunschweig (c3e) was extending, debugging and stabilizing the framework as a preparation to publish it to the project consortium with legwork and support by TU Dortmund. Please contact meyer@c3e.cs.tu-bs.de for more information.
In addition to that the following changes are made from D4.16:

- Deeper introspection via sr_report and USI
- Full implemented models of NoC and Network interface of IDAMC (in separated repository)
- ARM Cortex-A9 Processor Model available in three abstraction levels (loosely-timed, approximately-timed and cycle-accurate)
- Augmenting the LEON3 cache with a time-predictable coherence control algorithm, will work with the Cortex too
- AHB arbiter support for per-transaction priority arbitration
- Introducing a MicroBlaze Processor Integer Unit
- On-demand cache coherence algorithm, currently implemented for the LEON data cache
- Improved version of Trap-Gen to support ARM and MicroBlaze with register access via scripting
- Processor register models can be parametrized in runtime from untimed down to cycle-accurate/pipelined behavior
- GREth model (in separated repository)
- Better scripting integration
- Top-level instantiation in Python
- And many more (see commit log)

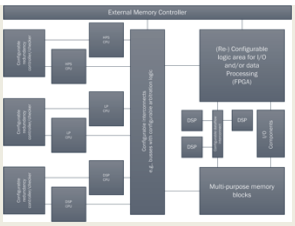
You can find our code and documentation at http://socrocket.github.io/

Download the core platform for EMC2 partners from https://github.com/socrocket/emc2. The code is available under AGPL. Additional repositories or license schemes are available. Please contact us:

Contact:
- TU Braunschweig, c3e, Rolf Meyer, meyer@c3e.cs.tu-bs.de, +49 531 391 2378
- TU Braunschweig, c3e, Jan Wagner, wagner@c3e.cs.tu-bs.de, +49 531 391 2387
- TU Dordmunt, Lilian Tadros, lillian.tadros@tu-dortmund.de, +49 231 755-4670

3. Asymmetric Multiprocessing on Zynq and HDMI I/O demonstrator

Technology lanes covered:
- **Heterogeneous Multiprocessor SoC architectures**
- **Dynamic reconfiguration on HW accelerators** *and reconfigurable logic*
- *Networking*
- *Virtualization and verification technologies*



3.1 Introduction

Video processing and very fast digital I/O requires processing based on combination of standard processors and HW acceleration blocks in programmable logic. The Xilinx 28nm Zynq device contains two 32bit ARM Cortex A9 processors and programmable logic on a single chip.

The Zynq device is complemented by DDR3 memory to form a system on module (SoM). Different SoM modules are fitted with different grades of the Zynq devices covering different speeds and temperature ranges. This is also reflected in different cost of the SoM modules. A Zynq SoM module is located on a carrier board with specific form factor and I/Os needed in the concrete application.

We describe set of UTIA HW accelerator designs developed for the EMC2-DP-V2 platform developed by Sundance within the EMC2 project. EMC2-DP-V2 is carrier working with the Trenz electronic TE0715-03-30-1I Zynq System on Module (SOM). See *Figure **5*** and also *Figure **26*** and *Figure **27***.

The Sundance EMC2-DP-V2 carrier board provides the PCI-Express connectivity, serving for construction of larger (stacked) parallel systems. The board is extended with an I/O card with FMC connector serving for input of Full HD HDMI video data and for output of Full HD video data.

Presented HW acceleration of video algorithms is using the Xilinx SDSoC 2015.4 development environment. The SDSoC is supporting development and debug of algorithms in C/C++ and compilation of selected C/C++ functions to HW accelerators. Accelerors are complemented with auto-generated data movers. The SDSoC environment needs board support package (BSP) with low level description of board I/Os and related HW IP cores. This BSP has been developed by UTIA for the EMC2-DP-V2 HW platform. See Chapter 9 for Sundance board and demo descripton.

The developed BSP (see *Figure **4***) includes the asymmetric multiprocessing demonstrator (AMP) platform with one MicroBlaze processor and three run-time reprogrammable (8xSIMD) EdkDSP floating point accelerators in the programmable logic part of Zynq device of the EMC2-DP-V2 platform (see *Figure **5***).

UTIA is providing application notes and evaluation packages presenting the AMP platform with Full HD HDMI, HW accelerated video processing, Full HD HDMI output. It is working on single Zynq chip in parallel with the three floating point accelerators coordinated by 32bit Microblaze processor working in parallel in asymmetric multiprocessing mode with the ARM Cortex A9 processor code.

3.2 AMP with three (8xSIMD) runtime reconfigurable EdkDSP accelerators

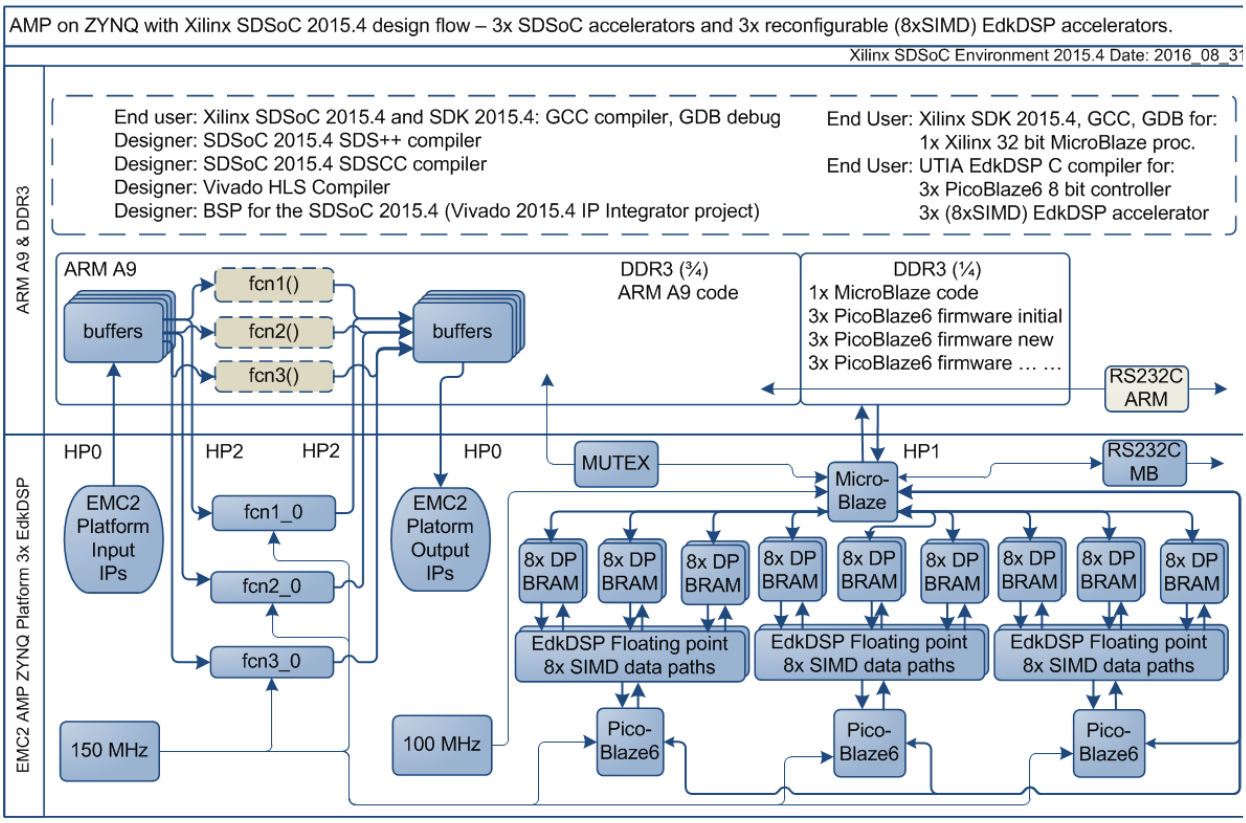


Figure 4: AMP design on EMC2-DP-V2 platform 3x EdkDSP@ 150MHz.

3.2.1 Innovation demonstrated

The BSP with AMP platform for the EMC2-DP-V2 is compatible with the Xilinx Software Defined System on Chip (SDSoC) 2015.4 environment [1],[2],[3],[4].

The AMP platform supports triple redundancy of 3x (8xSIMD) EdkDSP reconfigurable single precision floating point accelerators in parallel to the HW accelerated Full HD video-processing.

Key performance figures for each of the three (8xSIMD) EdkDSP reprogrammable accelerators:

- 1331 MFLOP/s each AMP accelerator (in case of FIR floating point filter)
- 891 MFLOP/s each AMP accelerator (in case of adaptive LMS floating point filter)
- Sobel filter based edge detection with 1, 2 or 3 three HW data paths in Full HD. or
- Motion detection with video processing pipeline of Sobel filters and median filter.

Support application notes and evaluation SW/HW packages for this EMC2-DP-V2 platform have been released for public download. See:

- SDSoC 2015.4 Standalone BSP with Full HD HDMI In-Out with SW and HW Demos for Zynq System-on-Module TE0715-03-30 and Sundance EMC2-DP-V2 Platform[2]

[2] http://sp.utia.cz/index.php?ids=results&id=s30i1h2

- Full HD HDMI In-Out HW-Accelerated Demos for Zynq System-on-Module TE0715-03-30-1I and Sundance EMC2-DP-V2 Platform[3]

- EMC²-DP HDMI in HDMI out Platform[4]

These released EMC2-DP-V2 platform application notes and evaluation packages are introduced in the next sections.

3.3 SDSoC 2015.4 Standalone BSP with Full HD HDMI In-Out with SW and HW Demos for Zynq System-on-Module TE0715-03-30 and Sundance EMC2-DP-V2 Platform

3.3.1 Key features

The released application note describes installation and use of a stand-alone board support package (BSP) for the Xilinx SDSoC 2015.4 for the Sundance EMC2-DP-V2 platform with commercial or industrial grade Zynq XC7Z030-1C or XC7Z030-1I device on System on Module TE0715-03-30-1C or TE0715-03-30-1I. The stand-alone BSP package includes 3 motion detection and 3 edge detection video processing demos [2].

3.3.2 Main objectives:

- To demonstrate how to install, compile, modify and use the BSP and SW projects in the Xilinx SDSoC 2015.4 for the Sundance EMC2-DP-V2 platform.
- To demonstrate on the EMC2-DP-V2 platform the HW accelerated video processing algorithms and the speedup against the initial SW versions running on the ARM.

3.4 Full HD HDMI In-Out HW-Accelerated Demos for Zynq System-on-Module TE0715-03-30-1I and Sundance EMC2-DP-V2 Platform

3.4.1 Key features

The released application note describes use of an evaluation package with 3 edge detection and 3 motion detection video processing designs on the Sundance EMC2-DP-V2 platform with industrial grade Zynq XC7Z030-1I device on System on Module TE0715-03-30-1I. Algorithms have been compiled by Xilinx SDSoC 2015.4 System level compiler (based on the LLVM framework and the Xilinx HLS compiler) to Vivado 2015.4 projects, and compiled by Vivado 2015.4 to bitstreams for the EMC2-DP-V2 platform. The SW access functions controlling the HW accelerators have been exported to the Xilinx SDK 2015.4 SW projects as static .a libraries for standalone ARM Cortex A9 applications. Application note also describes 3 edge detection algorithms coded in format supporting user defined ARM C code which is executing in parallel with the video processing HW paths.

3.4.2 Main objectives:

- To demonstrate the Sundance EMC2-DP-V2 platform.
- To demonstrate how to install, compile, modify and use the enclosed SW projects in the SDK 2015.4.
- To demonstrate the HW accelerated video processing algorithms and the speedup against SW versions.
- To demonstrate parallel processing of predefined video processing HW paths with C user code on ARM.

[3] http://sp.utia.cz/index.php?ids=results&id=s30i1h1
[4] http://sp.utia.cz/index.php?ids=results&id=emc2-dp-platform

3.5 EMC²-DP HDMI in HDMI out Platform

3.5.1 Key features

The released application note provides description of UTIA HDMI in HDMI out video demonstrator based on Sundance EMC2-DP carrier, Trenz System on Module SoM based on Xilinx Zynq and Avnet FMC extension card[13],[15]. The text describes standalone and Petalinux solutions.

3.5.2 Main objectives:

The HDMI in HDMI out video platform consists of the following components:

- Hardware built from Sundance EMC2-DP v2 carrier board, FPGA SoM Trenz TE0715-7030 and FMC IMAGEON extension card.
- Xilinx Vivado 2015.4 HW description file.
- Xilinx SDK workspace with HDMI in HDMI out application.
- Petalinux 2015.4 sources with BSP file prepared for the Vivado 2015.4 design.
- Precompiled SD card files for platform quick test.

The demonstrator provides standalone and Petalinux version of HDMI in HDMI out SW example. Both of them require Xilinx SDK 2015.4 tool to compile and debug them. The example is a simple video pass through demo using 3 frame buffers located in DDR3 memory.

3.5.3 Conditions for use

The application packages and evaluation designs are available for public free download from http://sp.utia.cz/index.php?ids=projects/emc2.

Contact:

- UTIA AV CR v.v.i. Jiri Kadlec, kadlec@utia.cas.cz, tel. +420 2 6605 2216

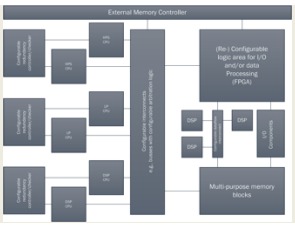
Figure 5: EMC2-DP-V2 platform with Zynq xc7z030-1I device and 3xEdkDSP accelerator

Figure 6: EMC2-DP-V2 Full HD HDMII-HDMIO Edge Detection and AMP on 3x EdkDSP

4. Dependable and high accuracy time transfer for distributed control applications

Technology lanes covered:

- *Heterogeneous Multiprocessor SoC architectures*
- *Dynamic reconfiguration on HW accelerators and reconfigurable logic*
- **Networking**
- *Virtualization and verification technologies*



Demonstrator presents a new concept for time and frequency dissemination over Ethernet networks. Our reference technology is the Precision Time Protocol v2 (PTP, IEEE-1588) modified and renamed as White Rabbit (WR). The technology was born at CERN and Seven Solutions is leading its utilization on industrial applications and, in the framework of EMC², its use for distributed safety critical applications with time or frequency as critical data. WR is the most accurate, flexible and the easiest solution for network synchronization over Ethernet. The main features of the WR technology are:

- Deterministic time transfer
- Best effort delivery for generic data
- Sub-nanosecond synchronization accuracy
- Hardware timestamps
- It is possible to interconnect thousands of nodes
- Typical distances of 10km between nodes (can be extended to more than 100km)
- Ethernet-based Gigabit rate reliable data transfer

More details can be found at: http://www.whiterabbitsolution.com/. Current WR high performance is possible thanks to the extension of well-known network standards, such as Synchronous Ethernet (SyncE) and PTP. In addition, the combination of them with several mechanisms to compensate link asymmetry, as well as monitoring and control clock phase information, form the pillars of this technology.

WR networks are mainly composed of two types of elements forming a tree-hierarchical model. These elements are switches and nodes. The demonstrator here described provides the deterministic time and frequency transfer capabilities previously mentioned together with redundancy mechanisms to increase dependability. These functionalities are developed in the framework of the EMC² project.

4.1 WR demonstrator features & elements

Redundancy, deterministic time distribution capabilities and safety aspects are achieved by the combination of several features, where the common notion of time is possible thanks to a very high time transfer accuracy that plays a key role. The main properties are:

- First, WR is used as the main technology to distribute the global time reference for large distances in a deterministic and dependable way with sub-nanosecond accuracy.
- Second, WR Switches and nodes produced by Seven Solutions implements redundancy protocols such as HSR for both timing and data providing timing networks with fault tolerance mechanisms.
- Third, deterministic features for time distribution are obtained thanks to the inclusion of QoS features, highly accurate latency measurements and mechanisms to guarantee packet delivery.
- Fourth, in contrast to the WR deterministic time dissemination, generic data transfer is performed following a best effort delivery approach.
- Fifth, Seven Solutions WR devices are also capable of disseminating time references using different technologies such as PTP, IRIG-B and NMEA, forming a heterogeneous, versatile and diverse time provider for Smart Grid.

The distributed control is based on a mechanism to distribute command actions through the network. Thanks to the common notion of time, the safe operation of the system relies on providing mechanisms to guarantee packet delivery on a bounded period of time. Those features are being developed for EMC².

Figure 7: WR-Switch, WR-ZEN (Zynq architecture) and WR-LEN (node based on Artix FPGA).

As previously said, the WR network elements are switches and nodes. For the EMC² project, we will focus on the WR-Switch, the WR-ZEN board and the WR-LEN. The WR-Switch has been a key component of WR technology composed of 18 SFPs developed as a Virtex-6 FPGA (XC6VLX240T) and a 400 MHz ARM processor (Atmel AT91SAM9G45). The WR-ZEN integrates the latest Xilinx Zynq Z-7015 device with a Dual ARM® Cortex™-A9 and containing an Artix FPGA-logic with 74K logic cells. The WR-ZEN has also been designed with an ultra-high stable oscillator and PLLs to provide a significant better short-term stability than previous WR boards. An external high-stability oscillator input is also available for dealing with situations that require a very high holdover. The combination of the hard-core dual processor with the Lattice LM32 soft-core require the utilization of safety-critical design techniques such as the ones addressed on WP4 and required by the certification standards like IEC-61508. Nevertheless, the focus of Seven Solutions on this project and the goal of this demonstrator are the inter-chip communication mechanisms here addressed as well as the development of fault tolerance mechanisms to avoid single point of failure increasing the dependability of time & frequency transfer in Smart Grid networks.

4.1.1 Time & Frequency Transfer Robustness

Avoiding single point of failure as single communication links is a necessity to be addressed in safety critical applications. Currently, WR robustness is implemented as a Rapid Spanning Tree protocol (RSTP) able to reconfigure the network whenever a node is down in the order of 30 ms, however, Seven Solutions (7S) is working on the integration of a redundancy protocol to achieve zero-time recovery in WR networks. The reference protocol is known as High-availability Seamless Redundancy (HSR, IEC 62439-3) and guarantees zero-time recovery in case of failure. By including the protocol in WR elements, we could extend HSR features to time and frequency distribution in WR ring networks. For this purpose, 7S is working on a HSR IP-Core that will be included in next-gen nodes to enable HSR configurations to guarantee that, in case of failure, both time and frequency will remain synchronized with no loss.

The elements shown in *Figure 8* combine WR nodes as WR-LENs, WR-ZENs and the WR switch to perform a deterministic time and frequency network using different time protocols (WR-PTP, IRIG-B, NMEA, etc.). In order to provide the network with robustness features, part of the time distribution network has been configured as a HSR ring able to avoid single point of failure with zero-time recovery. It is also possible to group other ring networks by using the WR-ZEN (or WR-Switch) as a Redbox or a Quadbox (with proper expansion FMC board).

The target demonstrator combines all these devices to illustrate the time and frequency transfer accuracy and the possibility of using different technologies for time distribution as well as the implementation of redundancy methods and QoS to meet safety requirements. Moreover, different

interfaces such as, PTP, IRIG-B, NMEA, 10MHz/PPS are available to ease the interoperability of different type of devices in Smart Grid networks.

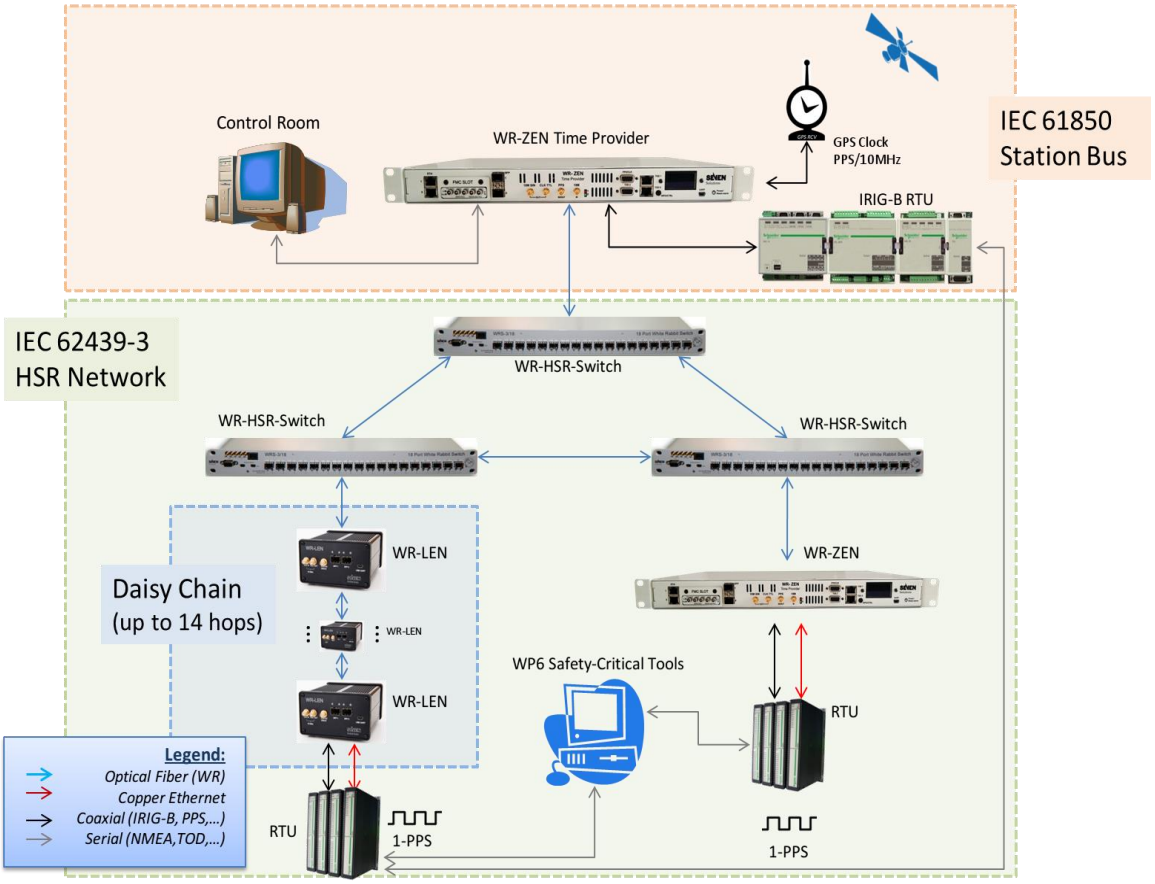


Figure 8: Heterogeneous Time & Frequency Distribution network using White-Rabbit (PTP) and IRIG-B with HSR capabilities to avoid single point of failure (WP11 LL Internet of Things UC 5.5).

4.2 Conditions for use, downloads

This demonstrator is provided by Seven Solutions EMC² partner. Concept, publication and reports will be available for EMC² partners via EmDesk and public information will be accessible at Seven Solutions website: http://www.sevensols.com/index.php. Electronics boards are commercially available and can be ordered through the company sales department. For concrete usage conditions and access rights please use contact persons info below.

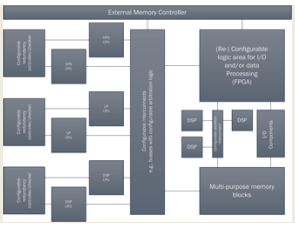
Contact:

- Seven Solutions S.L Javier Diaz, javier@sevensols.com
- Seven Solutions S.L Benoit Rat, benoit@sevensols.com
- Seven Solutions S.L Jos Luis Gutierrez, jlgutierrez@sevensols.com

5. Automotive domain: Network technologies and connection to testbed (based on UC7.3)

Technology lanes covered:

- *Heterogeneous Multiprocessor SoC architectures*
- *Dynamic reconfiguration on HW accelerators and reconfigurable logic*
- **Networking**
- **Virtualization and verification technologies**



The goal of the use case T7.3 "*Design and validation of next generation hybrid powertrain / E-Drive*" is the tailoring and further enhancement of the EMC² technologies for the design and validation of next generation hybrid powertrains and e-Drives. The validation part draws upon technologies from WP4, to implement and study high-performance connectivity between two dissimilar multi-core computation systems (e.g. xCU and V&V system). This use case addresses the following topics (business needs):

1. AVL_BN_007: Networking solutions for heterogeneous automotive systems.
2. AVL_BN_008: Tools and Methods to handle configuration of complex multicore-systems during development, calibration and diagnosis

As described in previous deliverables, multicore innovations developed within EMC2 are applied to implement complex, networked, powerful control units and systems thereof, for the next generation cars, trucks, farming vehicles.

However, such smart functionalities require sophisticated tools for testing (i.e. verification and validation, V&V) like

- Detailed, real-time simulation of driver, components, subsystems, environment, …
- Communication (IVN, V2x, …)
- Closed loop control with high dynamics
- Instrumentation & measurement systems
- Automation of testing
- Model-based exploration, optimization, testing
- Online processing and validation of results
- Deal with future variability and complexity
- Functional safety tests

Similar to the units under test (xCU's), the V&V system itself will be a high-performance, dependable computation platform, based on modern multi-core architecture. However, as significantly more memory, performance, throughput etc. are required, these platforms use typically industry standard x86/x84 platforms, with 8 and more cores in parallel.

Therefore, the V&V system will have to support

- Highly scale-able and configure-able systems → adapt-able
- Multiple applications based on (standardized and specific) SW components
- Providing real-time execution with typical cycle rates of 50us and above
- Based on COTS platform: the industry PC (multicore x86/x64)
- Allow (automatic) partitioning and thus parallelization of applications

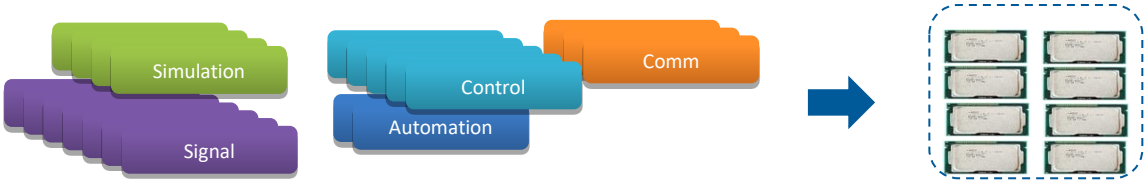


Figure 9: Testing complex xCU's: multiple test tasks of the V&V system communicate with multi-core xCU

In order to execute testing, a highly performant, dependable, deterministic communication link between UUT and V&V system is required.
It must support standard IVN (in-vehicle network) implementations, i.e.

- CAN
- CAN-FD and/or
- Automotive Ethernet

New approaches allow managing consistency, variability, authenticity of data and its structure in such a highly parallelized system within the constraints of a vehicle's hardware.

This demonstrator will show solutions for configuration, management, communication, verification, and the required hardware connectivity for performant data exchange for the described scenarios.

Multiple tasks running on multicore xCU's will communicate with multiple tasks executing on V&V systems. This is implemented via unified high-performance, low latency real-time networks

In phase 1, requirement engineering, concept design and base architecture has been developed. In phase 2, a first version of the demonstrator with a high performance test application interfacing to one or more of the multi-core xCU's developed by partners was shown, see Figure 10: *Testing xCU apps with a complex V&V system*. In the third phase, a second iteration of the system shall demonstrate the full potential of the technology.

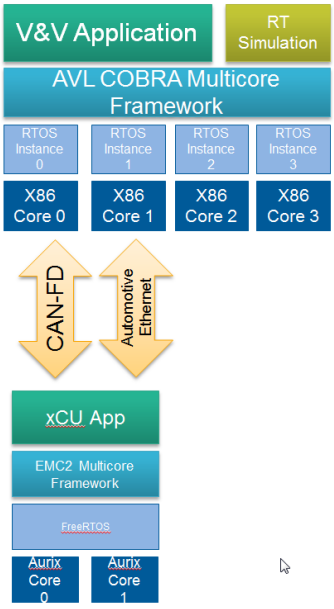


Figure 10: Overview: Testing xCU apps with a complex V&V system

This demonstrator will therefore allow showing and analyzing efficient validation of next generation hybrid powertrain / E-Drive systems in combination with powerful multi-core based real-time test & simulation tools via multiple logical connections / streams through high-performance automotive networks.

5.1 System description, parameters, photos, diagrams

The demonstrator is based on the basic layout depicted in Figure 11 (note: several alternatives for the RTOS on XCU side are possible)

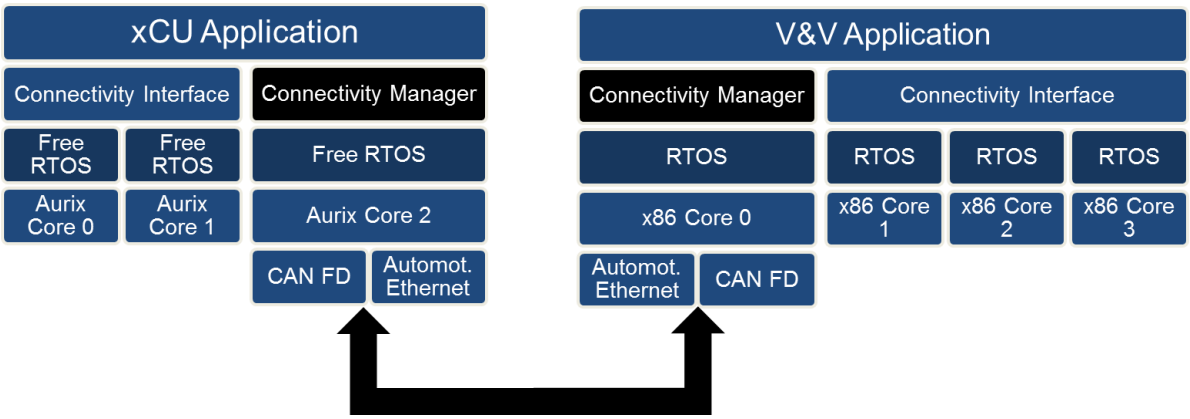


Figure 11: Communication architecture from xCU to V&V system

The demonstrator will be implemented as part of an automotive test bed (e.g. vehicle or engine based, see Figure 12 and
Figure *13*).

Figure 12: Vehicle Test Bed

Figure 13: Engine Test Bed

5.1.1 Concept

The developed EMC² demonstrator consists of two components in order to ensure multicore capability. As shown in Figure 14 there is one main component ("Connectivity Manager") executed on a predefined core. On every additional core, an instance of the Connectivity Interface is executed, which takes care of the inter-core communication.

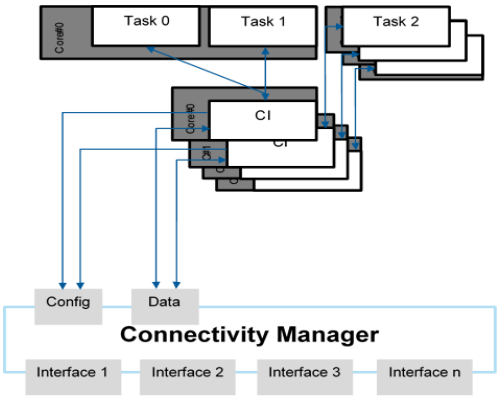


Figure 14: Software Concept of the demonstrator

The Connectivity Manager is in control of the hardware interface (e.g. CAN Interface) and is responsible for prioritizing and interlocking messages. Moreover, the Connectivity Manager is associated with several Connectivity Interfaces, whereas one Connectivity Interface may serve one or more several tasks. All information streams are bundled at the Connectivity Manager and prioritized according to the **criticality of the task**. Before a new information stream registers at the Connectivity Manager, a bandwidth check is performed in order to ensure that there is enough bandwidth available to handle all registered information streams. This assures that accepted information streams are able to transmit their data within a given cycle time. A sophisticated algorithm takes care of the interlocking of the messages based on an earliest deadline first concept. In case of additional load on the bus, the algorithm adapts to the remaining bandwidth and changes the transmission order so that information streams with a high criticality are favored over information streams with a lower criticality. By the time that nominal bandwidth is available again, the Connectivity Manager tries to make up the delay by giving additional bandwidth to delayed information streams while keeping the high critical information streams unaffected.

5.1.2 **Results**

The well designed concept yields for the following advantages:
Multiple communication partners can independently use the same communication link, also known as multiplexing. Whereas the information streams are interlocked so that

- the real-time requirements of the communication partner are optimally considered
- the Connectivity Manager can adopt to a temporary bottleneck at the bus (e.g. reduced available bandwidth due to additional work load on the bus)
- the criticality of each information stream is taken into consideration in case of real-time violations. Higher critical information streams are favored over lower critical information streams.
- there is only a short processing time needed to prioritize and transmit data.

Moreover, the system performs a bandwidth-check before a connection is registered; this ensures on the one hand side that there is enough bandwidth available for all registered connections and on the other hand side that the capacity of the shared link is not exceeded.

The system has been tested in a close to real life scenario:

5.1.2.1 Scenario

- 5 Connections from two INtime Nodes (multicore)
- Various criticality, same cycle time and data volume
- > 100% of bandwidth utilization with temporary additional work load
    - Proof of working algorithm (prioritization of higher critical connections)
- 4 % of RT Kernel CPU usage in stress test (5 % peak at initialization)

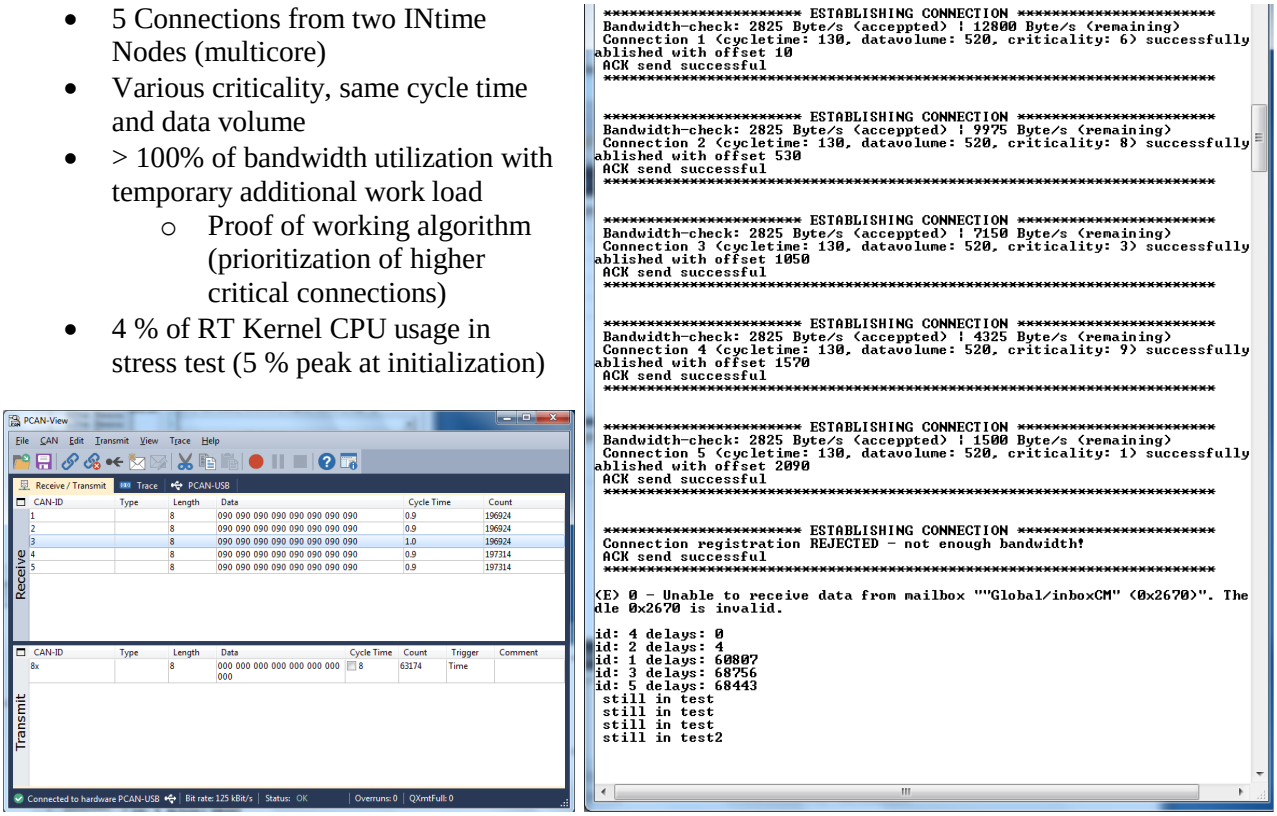


Figure 15: Prioritization of information streams

Figure 16: Received CAN messages

5.1.2.2 Other scenarios:

- Long term test (~659M successfully transmitted CAN messages in 72 hours)
- Bandwidth test (99% of bandwidth utilization over 1,5 hours without delays)

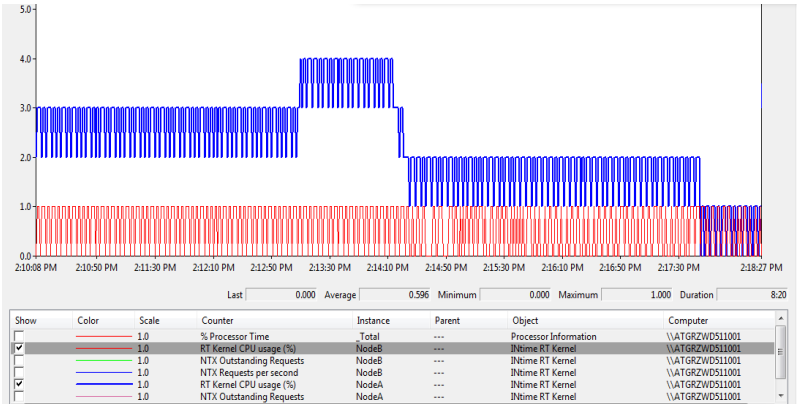


Figure 17: Minor increase of CPU usage

5.1.3 Conclusion

- Prioritization of information streams across two cores (multicore)
- Bandwidth utilization up to 99%
- Stable system (72 hours test)
- Minor increase of RT Kernel CPU usage

5.2 Conditions for use, downloads

This demonstrator is provided by partners AVL/GRZ, EB, IFAG.
Concept, publication and reports will be made available for EMC2 partners via EmDesk.

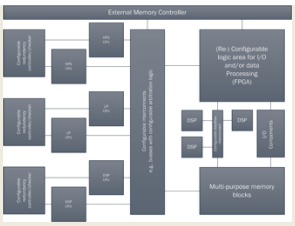
For re-use and additional information, please contact one of the following persons:

- AVL/GRZ, Eric Armengaud, eric.armengaud@avl.com;
- AVL/GRZ, Peter Priller, peter.priller@avl.com
- EB, Alexander Mattausch, alexander.mattausch@elektrobit.com
- IFAG, Werner Weber, Werner.Weber@infineon.com
- VIF, Allan Tengg, allan.tengg@v2c2.at

6. Time-triggered Multi-processor System-on-Chip Architecture for Cross-domain applications

Technology lanes covered:

- **Heterogeneous Multiprocessor SoC architectures**
- *Dynamic reconfiguration on HW accelerators and reconfigurable logic*
- *Networking*
- **Virtualization and verification technologies**



6.1 EMC² TTNoC Many-core Architecture on Altera Arria V

A time-triggered many-core architecture has been proposed as a viable candidate to solve the problem of the migration of safety-critical systems and mixed-criticality systems from single-core to multi-core processors. An FPGA based platform has been used to implement first prototype called ACROSS MPSoC[17]. It was a significant achievement that showed how one can increase performance of a system while keeping a resilience factor of the system on the level required by the authorities. In order to achieve the performance levels closer to the ones of state-of-the-art multi-core processors we proposed to implement the same architecture on a hybrid system-on-chip (e.g., Arria SoC, Zynq SoC).

With support of TTTech, TUW successfully ported time-triggered MPSoC architecture to the to the hybrid system-on-chip platform Arria V, developed by Altera (see *Figure 18*). The hybrid SoC combines an FPGA device with a hard-coded single- or multi-core processor, in this case Arria V FPGA and ARM Cortex C9 processor. Two devices are connected via multi-channel on-chip interface that provide efficient and fast communication between them. The interface allows devices to share interaction with each other during a boot process, and to share resources (e.g., memory, peripherals). The vocal part of the architecture is again time-triggered network on chip (TTNoC), provided by TTTech. The original TTNoC needed to be adapted and ported to the new platform, and current development toolchain. The demonstrated system contains five Nios II soft-coded cores integrated in individual computational units, completely isolated in space and time, and interfaced with the TTNoC. The architecture also integrates ARM core, as an isolated component with connection to the TTNoC. *Figure 19* shows a block diagram of the proposed architecture with possibility to expand it with additional components.

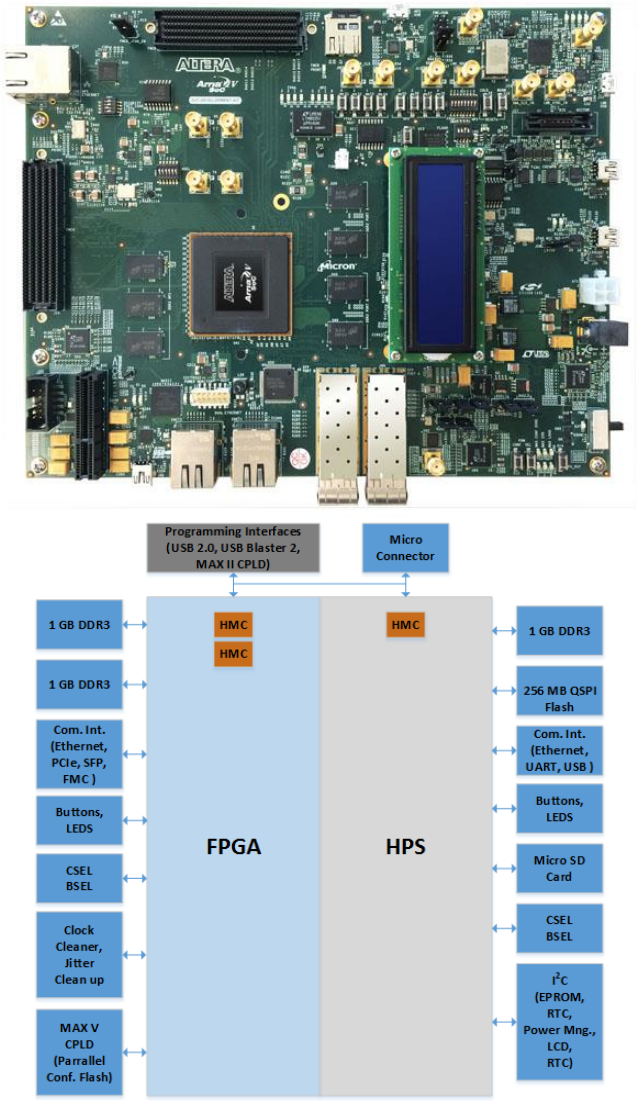


Figure 18 Arria V SoC Development Board and Architecture

The components implemented on the prototype are generic components and represent a base for more specialized and dedicated components. The architecture was initially designed to increase performance potential of the system by adding hard-coded processors in the mix. It would also be a proof of concept for the integration of heterogeneous components. However, the work on the hybrid SoC yielded a number of other potential advantages for the system, e.g., fault tolerance mechanisms, or dynamic reconfiguration and adaptation.

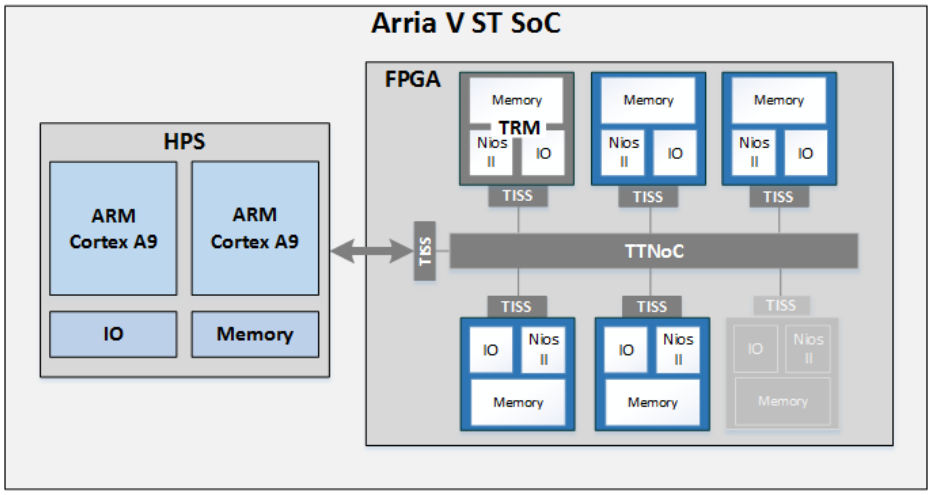


Figure 19 Heterogeneous time-triggered architecture based on hybrid SoC platform

The hybrid SoC platforms allow mutual control of the devices, which includes boot up, reset and shut down. This is especially interesting for MPSoC architectures, as initially each component was implemented in spatial and temporal isolation. This would allow each component to recover from a fault. In case of an error where the whole FPGA was affected the containment unit was chip or device. This would result in the whole system being affected. The structure of the hybrid SoC allows the system to recover or repair one of the components like a physically independent device. They are still implemented on a same chip but could operate completely independent from each other.

The implementation of such architecture is not limited on Arria V devices. The core blocks of TTNoC can be implemented on other hybrid SoC platforms, i.e. Xilinx Zynq. The capability of heterogeneous integration of the TTNoC allows implementation of components based on other soft-coded CPUs (e.g., Leon 3), hard-coded CPUs or dedicated cores (i.e., accelerators, DSP).

Further information of the architecture can be found in the project deliverables D4.3, D4.4, D4.16. The work on the architecture described is published on ISIE 2016 [23]

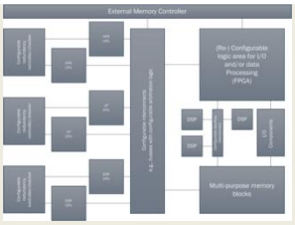
6.2 Conditions for use, downloads

This demonstrator is provided by TUW, with support of TTTech, who are owners of intellectual rights for certain IP modules in the architecture (e.g., TTNoC). For the support in integration and usage the contact person is Haris Isakovic (email: haris@vmars.tuwien.ac.at) from TUW, for further information acquiring rights to specific components the contact person is Andreas Eckel (email: andreas.eckel@tttech.com) from TTTech.

7. Enhanced Automotive Time-of-Flight Demonstrator

Technology lanes covered:

- **Heterogeneous Multiprocessor SoC architectures**
- *Dynamic reconfiguration on HW accelerators and reconfigurable logic*
- *Networking*
- *Virtualization and verification technologies*



7.1 Introduction

Indirect Time-of-Flight is a very suitable technology for environmental perception in an automotive environment, as outlined in the previous deliverables D4.15 and D4.16. In particular, thanks to its monocular principle, it can be integrated into small embedded devices.

During the EMC2 project, the automotive Time-of-Flight demonstrator has been developed in three major steps:

- A very first demonstrator, based on the AURIX automotive platform, was shown in D4.15.
- In D4.16, a first version of the enhanced, i.e. hardware-accelerated, solution was introduced.
- Finally, while D4.17 presents an overview of the final Time-of-Flight demonstrator platform, D4.22 will contain detailed conceptual, design, implementation and evaluation information.

7.2 Final Time-of-Flight demonstrator platform

*Figure **20*** illustrates the final Time-of-Flight demonstrator assembly. It is made of Infineon Technologies' REAL3™ Time-of-Flight evaluation kit, a Xilinx Zynq 7020 FPGA, and the AURIX TC299 automotive computation platform.

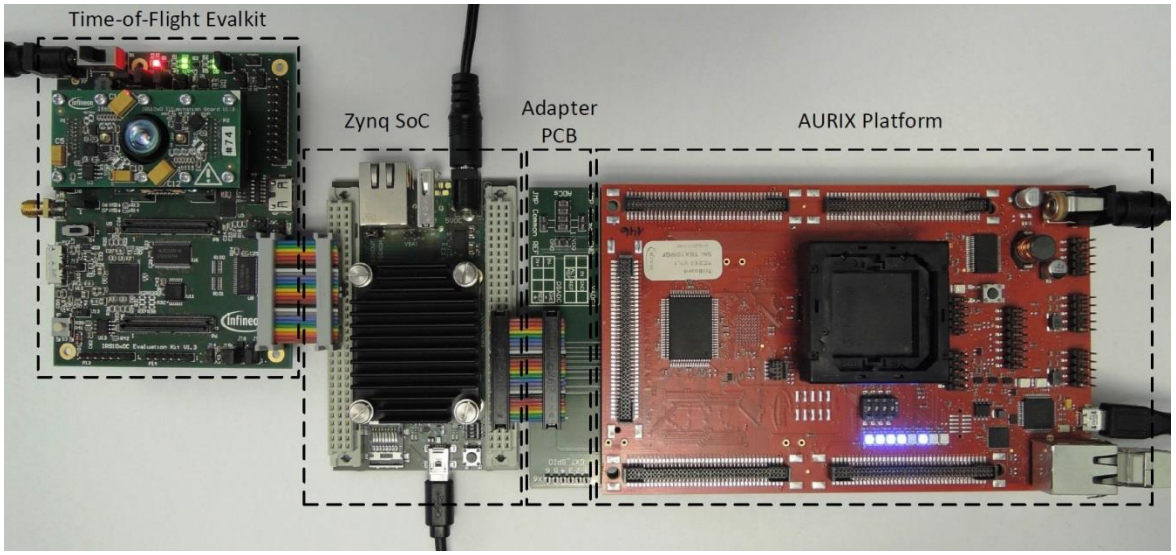


Figure 20: Final hardware-accelerated Time-of-Flight demonstrator platform

Figure 21 shows the basic flow of operations: the AURIX micro controller acts as the system's control unit. Its use-case application triggers the Zynq FPGA by means of a communication protocol. Consequently, the FPGA starts its software / hardware components and the Time-of-Flight camera system which provides the raw sensor data. The FPGA fetches the raw sensor data and performs the hardware-accelerated raw to depth data processing. Furthermore, computation intense use-case specific tasks can be preprocessed within the FPGA in order to reduce the performance and memory requirements of the AURIX micro controller for the use-case application.

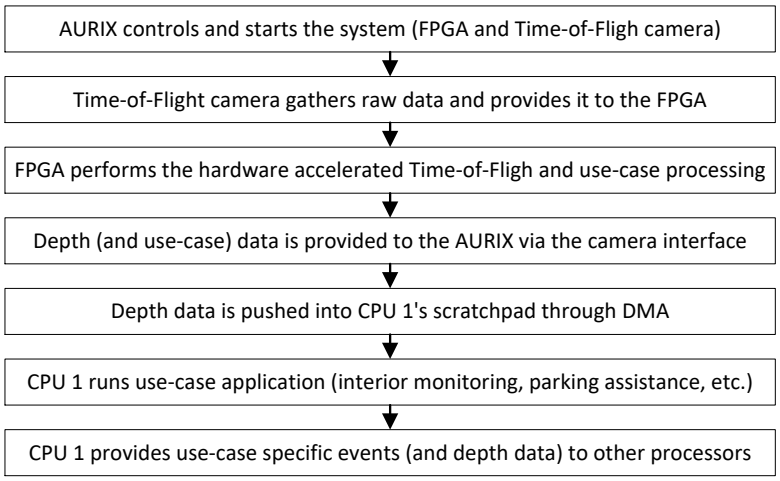


Figure 21: Flow of operations of the final Time-of-Flight demonstrator platform

Then, the processed Time-of-Flight depth and amplitude data and, if applicable, pre-processed application data is forwarded to the AURIX. For this purpose various interfaces (depending on the required data rate) can be used, such as a parallel camera interface, Ethernet, CAN bus, or I2C bus. The depth and amplitude data is received and then processed by the AURIX in a multi-core environment that supports mixed-critical and real-time applications. While one processor core executes the use-case application (e.g., background-replacement, interior monitoring) in an isolated way, the remaining cores are free for other tasks (e.g., lock-step mode for further functional safety aspects). During the execution of the use-case, dedicated status and control events are generated. Finally, the data (which may consist of depth data, amplitude data, 3D point cloud, and use-case specific events) can be forwarded by CPU 1 to other processing units for, e.g., displaying purposes.
Thanks to this hardware/software setup, hardware-acceleration and multi-core techniques are exploited and mixed-critical use-case applications are supported.

7.3 Conditions for use, downloads

This demonstrator is provided by partner IFAT.

For re-use or download please contact one of the following persons.

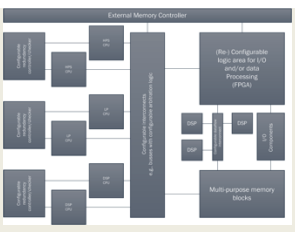
Contacts:

- IFAT, Thomas Herndl, thomas.herndl@infineon.com, tel. +43 (5) 1777 11572
- IFAT, Norbert Druml, norbert.druml@infineon.com, tel. +43 (5) 1777 5155

8. Fine Grain Partially Reconfigurable Array of Processors on Zynq with Linux reprogramming support

Technology lanes covered:

- **Heterogeneous Multiprocessor SoC architectures**
- **Dynamic reconfiguration on HW accelerators and reconfigurable logic**
- *Networking*
- *Virtualization and verification technologies*



This is a demonstrator built for Situational Awareness Partial Reconfiguration support. A typical application area is in Automotive Driver Assist, where the mixed criticality safety features are chosen based on real-time GPS, traffic type, roadwork and weather information. The required safety feature set implemented in FPGA firmware is supported by partial reconfiguration of the FPGA whilst maintaining a base set of safety related acceleration support. A User Space application running in Linux can request a partial reconfiguration through a Device Driver that will in-turn reconfigure the array processing subsystem. The reconfiguration is designed to switch the required safety features, thereby maximizing efficiency and minimizing power.

8.1 Description, parameters, photos, diagrams

The demonstration system will be a variation of EnSilica's Zynq eSi-Module and carrier board. The carrier board will allow interfacing to GPS, a Radar board, Ethernet for internet access, etc.

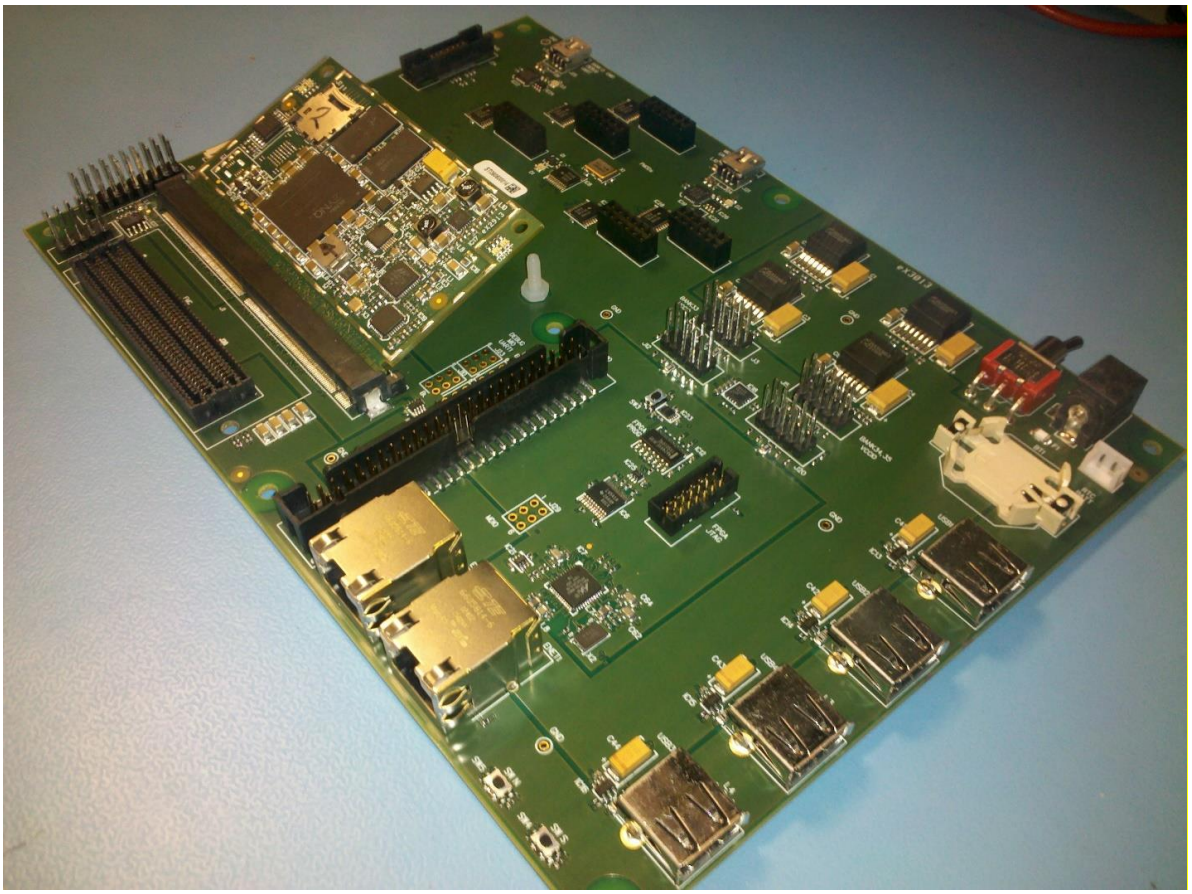


Figure 22: Photo of demonstrator carrier board and module

We are building a complete 4-channel ADAS processing subsytem in hardware. This includes the Range, Doppler, Azimuth and Elevation Fourier Transforms plus detection. Following the detection is a hardware subsystem that performs measurement association to tracks and then smoothing those tracks with an Extended Kalman Filter.

So far, we have been profiling ADAS signal processing IP components - FFT, CFAR, Tracking, Kalman Filter to determine the possible levels of redundancy and re-use that can be afforded in ADAS applications that cover long range RADAR (LRR) and short range RADAR (SRR). The intent is to determine a universal RADAR platform that meets the diverse needs of both LRR and SRR and can switch between the two application areas whilst maintaining sufficient redundancy to meet safety critical requirements. The research has looked at partial reconfiguration of the FPGA fabric to deliver that redundancy in time during the switch between LRR and SRR.

Next steps are to connect with an actual RADAR transceiver board and examine the real time processing requirements and the control of these from the application layer code.

To support the real-time demonstration we are enhancing the eSi-ZM1 module to use a Zynq Z-035 device, including a camera interface for real-time image capture, PCIe to store 30 minutes of image data on SSD and HDMI display interface.

8.2 Conditions for use, downloads

This demonstrator is provided by partner ENSILICA.

Contact: ENSILICA, Contact person, david.wheeler@ensilica.com, tel: 0118 321 7332

9. HDMI-in video processing to HDMI-out controlled by Ethernet port

Sundance has developed the EMC²-DP, a PCIe/104 OneBank™ board with dual ARM9 CPU, a re-configurable FPGA Logic and an interface to CPU specific I/O features[14]. The dual ARM processor cores have direct access to the DDR3 memory that provides 1GByte of storage.
The EMC²-DP can be used as either a Host Controller in a PC/104 Stack or as stand-alone as it is the case in this demonstration.

Demonstrator 8 is presenting the EMC² Development Platform controlled by a Graphical User Interface host application over an Ethernet connexion for live video. The Ethernet interface is made available on an add-on board called SEIC (Sundance External Interface Connector).

The purpose of this demo is to allow real-life data, in this case a video-stream from a HDMI Output, to be loaded into the Zynq's DDR memory and then displayed again on a second HDMI Input device (typically a monitor). In this example, a Xilinx 32-bit MicroBlaze CPU controls the transfer of data between the HDMI input, the DDR3 memory and the HDMI output.
The platform will allow real-time manipulation of the image, using either a number of CPUs or the FPGA fabric. There is no attempt in this demo to provide such features.

9.1 Description, photos, diagrams

In this demonstration, a VITA57.1 FMC® compatible Daughter Cards is plugged to the EMC²-DP to provide HDMI input/output capabilities. The input video is stored to DDR3 memory and the output video is read from DDR3 memory. A MicroBlaze 32-bit soft-core processor is implemented in the Zynq PL to control the HDMI interface to have access to the DDR3 memory and the video data for processing.

FIFOs are used to buffer the HDMI input and HDMI output to sustain HDMI rate. The HDMI output runs at the same time as the HDMI input with the same control signals delayed. The input and output HDMI rate are identical.
The memory read and write are alternated to provide data for HDMI output while storing the HDMI input data. The memory read is performed ahead of the memory write.

The output resolution is set at 1280x720 and the video frequency is set at 65MHz

The GUI host application controls the EMC²-DP via the 1GB Ethernet port using a write/read API. The Ethernet link is used to communicate with the ARM processor core running Linux that, in turn, communicates with the MicroBlaze processor to configure the system and drive its memory accesses remotely.

The host application can also provide any data to write to memory to be processed and then read it back. For example, it can load a picture to display on the HDMI output and read a picture captured on the HDMI input.

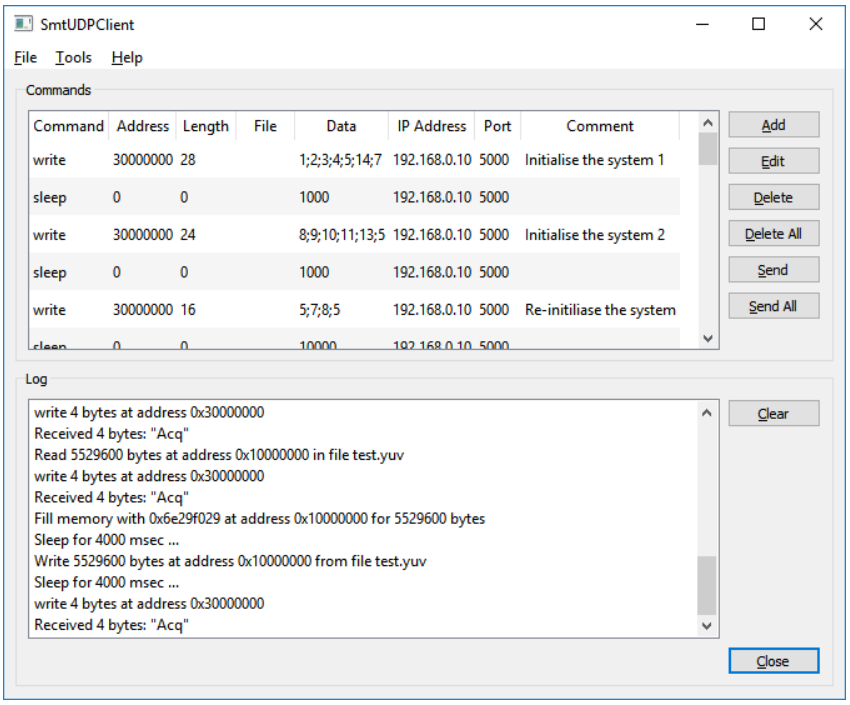


Figure 23: UDP Client GUI

A UDP server runs in Linux on the PS and the host application runs the UDP client. Some addresses in the DDR memory have been reserved for specific purposes. The UDP server on the PS and the Microblaze on the PL communicates with each other using these addresses.

The Xilinx® LogiCORE™ IP AXI VDMA core provides the high-bandwidth direct memory access between the DDR memory and the HDMI peripherals. The MicroBlaze controls the VDMA IP core to perform the read and write operations

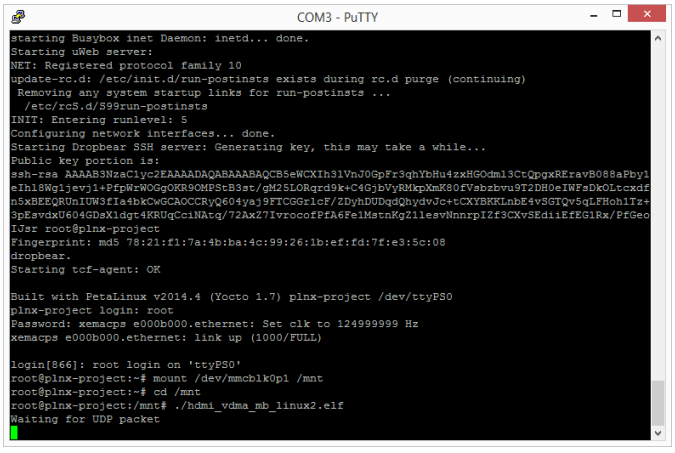


Figure 24: Linux server run

requested by the host. The MicroBlaze controls the HDMI input/output and reads their settings by posting a request to the right register. The software for the MicroBlaze runs as a bare-metal application and has no external dependencies.

The EMC²-DP boots from SD card so the application code would run at power up and could be used to configure the system from Ethernet.

From the graphic interface, the control commands can be sent manually. But more efficiently, the host application can load a set of commands from an xml file and send it to the system. Typically, a set of commands will be:

- read from memory (with the option to display data in the GUI and/ or save it to a file),
- write to memory (the data can be manually entered or come from a file),
- sleep.

An XML command file can also be created or modified from the graphic interface.

The use of XML files facilitates the system configuration but also makes it easier and quicker to repeat operations like running full system tests. Therefore the system is very flexible and can be controlled remotely using the Ethernet port.

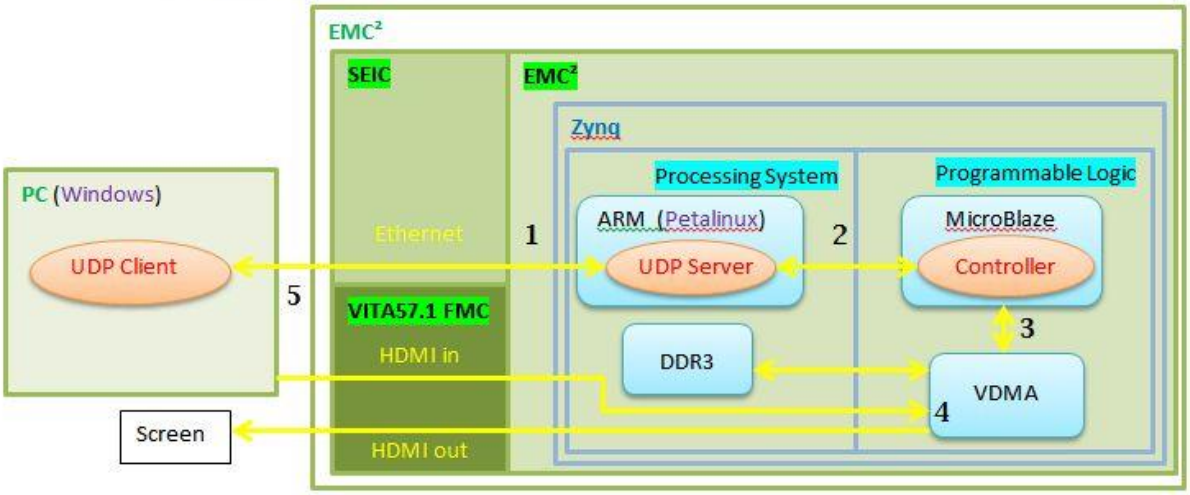
The embedded system operates under the control of a windows GUI by sending commands to start the acquisition on the HDMI input and display data on the HDMI output.
Thus this system consists of 4 main software/firmware parts:

- The firmware for the Zynq PL (Programmable Logic)
- The MicroBlaze Zynq PL.
- The UDP server in Petalinux on the Zynq PS (Processing System).
- The UDP client in shape of a GUI in Windows.

The GUI host application has been developed in C++ in the QT5 [5] [6] software environment to make the deployment to other platforms possible. The current application runs on Windows 8.

The firmware for the EMC² Zynq has been developed using the IP integrator Vivado 2015.2. Xilinx SDK was used to develop the ARM software. Petalinux 2014.4 is running on the ARM.

The figure below illustrates the system data flow.



1. The UDP Client sends a request to the UDP server via Ethernet.
2. The UDP server relays the request to the MicroBlaze.
3. The MicroBlaze controls the VDMA IP core to perform the request.
4. The VDMA IP core initializes and setups the HDMI input and output according to the MicroBlaze's commands.
5. The result of the request is sent back to the UDP Client.

Figure 25: System data flow

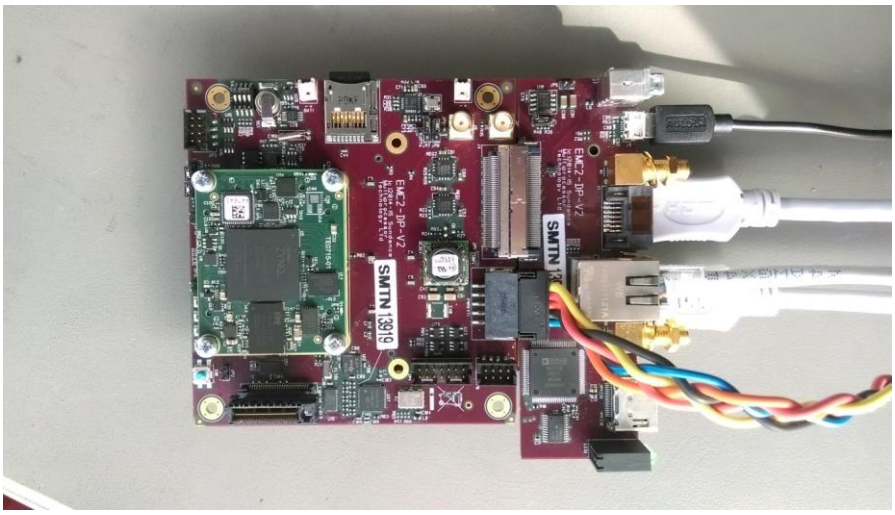


Figure 26: Diagram describing the demonstrator 8

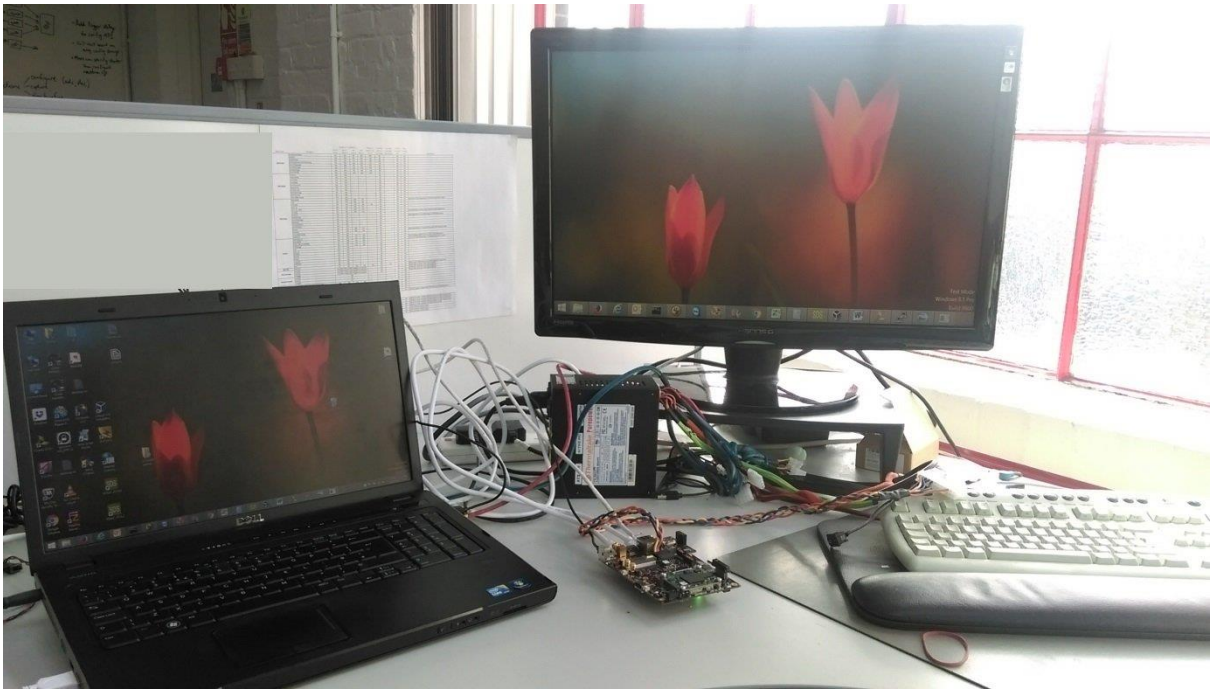


Figure 27: Photos of demonstrator 8

9.2 Conditions for use, downloads

This demonstrator is using Hackaday.io for its baseline support. Any Partner can get access to the EMC²-DP demo on the following website:
https://hackaday.io/project/12536-hdmi-in-to-hdmi-out-on-a-zynq-based-platform
where all the sources and documentations are available.

Contact: Sundance Multiprocessor Technology ltd., Emilie Wheatley, emilie.w@sundance.com

10.Demonstrator platform for multicore avionic systems

10.1 Description of the architecture

POLITO developed a proof-of-concept (PoC) demonstrator based on input coming from the Avionic Living Lab (WP8). The main goal of the PoC is the consolidation of several different applications with different levels of criticality on the same multi-core processor, where the main concern is the availability of a safety critical application. In order to grant application isolation, we used a type-1 hypervisor (HV): each application runs in its own partition; as such it has its own virtual memory space and its own channel to access peripherals; the HV guarantees isolation: it detects and stops any attempt of one application running in its own partition to access resources belonging to another partition. The architecture is presented in Fig. 1. The proposed architecture foresees the availability of a dual core processor, a Watchdog Timer (WDT), which should be able to reset the system in the event it becomes unresponsive, and one Watchdog Processor (WDP) one for each application, able to detect applications control flow errors (CFEs). The target system is a SoPC, and the programmable logic available in the system has been used to implement the WDs, which were custom designed.

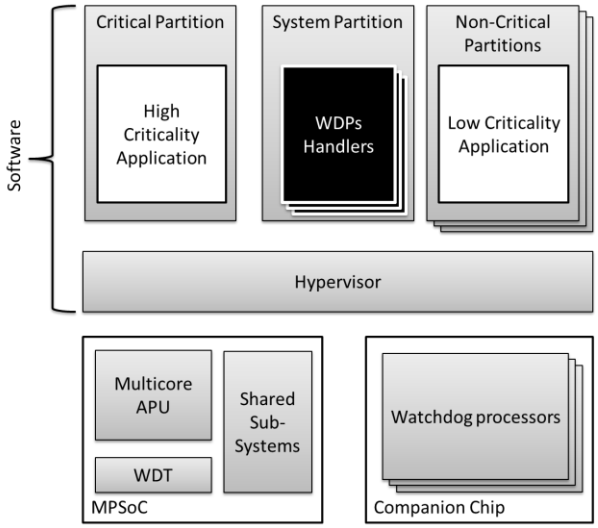


Figure 28. Proposed Architecture overview

As far as the software architecture is considered, the HV allows to:

- Isolate the applications by guaranteeing that an application can access only its resources. System resources are partitioned by the designer, and each partition is statically allocated to one application, which owns the resources exclusively.
- Provide support for restarting any partition whose WDP has detected a CFE.

The software architecture is composed of these partitions:

1. A critical partition (CP) containing one critical application (CA).
2. Several non-critical partitions (NCPs) containing non-critical applications (NCAs).
3. A system partition (SP) in charge of performing recovery actions (RAs).

The cooperation of these partitions among themselves and with the hardware is visualized in *Figure 29*. The HV is in charge of loading the images of the two applications and of instantiating the partitions. Once the system is up each application is responsible for configuring and enabling its own WDP. The critical application is responsible for refreshing the WDT, which in case of expiration resets the system. The WDT is responsible for recovering from faults preventing the execution of other recovery actions, e.g. single event functional interruptions (SEFIs). The non-critical applications should never interfere with the critical application, therefore only the latter can refresh the WDT, i.e. a system reset happens only when a fault leads to halting the critical application.
WDPs are used to detect CFEs; each application partition has a dedicated WDP. The application software is partitioned in blocks, each associated with a keyword. The blocks are executed sequentially and therefore any given WDP expects to receive signatures according to a predefined

sequence. Each WDP stores compile-time signatures, defined by the designer, in its own registers; the number of signatures in the sequence to be monitored by the WDP is also stored in a register. WDPs implementation is application independent: the configuration phase, in which the expected sequence is loaded in each WDP should be performed at bootstrap. Moreover, the time between one signature and the following one is configurable and is controlled by a timer. The timer is automatically reinitialized with the value stored in a specific register each time a correct signature is received. The application is responsible for sending the signature of each block to its own WDP. Each WDP triggers a dedicated interrupt if:

- It has not been enabled and the software sends a keyword;
- It has been enabled and the software performs any operation on it except sending a keyword;
- It receives a keyword different from the expected one;

Each of the listed conditions corresponds to a control flow error or a data error.

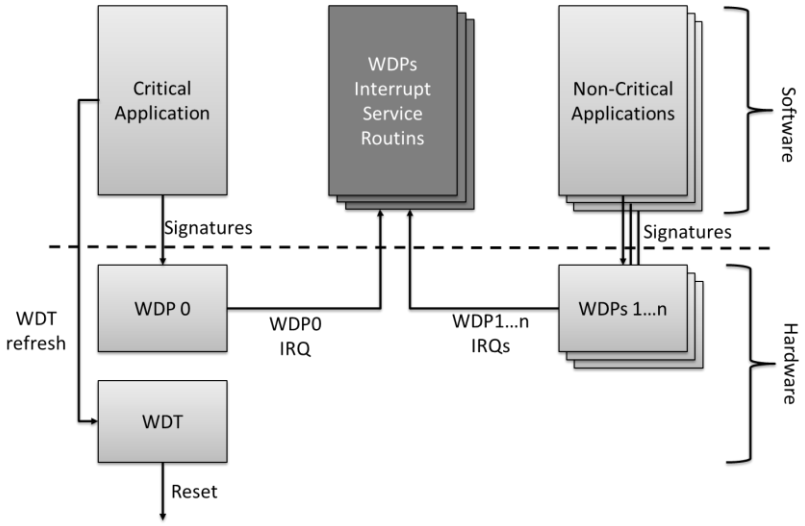


Figure 29. Messages exchanged among architecture components

The architecture implements multiple level of checks. First, the HV ensures that reserved resources are not improperly used by the applications running on the computer. This is mainly implemented through a separation mechanism, which depends on the HV implementation, and through a time partitioning scheduler. The latter grants that the tasks composing each application are allowed the proper time to execute, which is determined by the designer through a worst-case execution time analysis. In case one of this two mechanisms fails, the WDPs can detect the problem and allow for a fast recovery action, which is application dependent. In case WDPs are not able to detect or in case the system is in such a state that a fast recovery action is not possible, the WDT is able to detect the situation and triggers a secondary recovery action. Recovery actions are dependent on the application. In avionics applications, requiring a very high availability, the recovery action is often implemented through the use of a hot-standby spare computer: when a fault is detected in a computer, its hot-standby spare is activated. The crew is warned of such event, and the proper maintenance actions are taken in a timely fashion. The attentive reader may have noticed that the proposed architecture does not allow to avoid the use of spare computers, however it does allow to reduce the number of computers needed by the avionics, allowing to implement several applications on a single computer, ideally one application per CPU core.

Performance counters are available on most multicore architectures. Their intended use is to monitor performances of the system during the profiling and Worst Case Execution Time analysis phases. In the scope of this project, performance counters are used to perform online detection of temporal interferences due to software or hardware faults in the system which could cause deadline misses in some monitored tasks.

The proposed approach is to use the performance counters to control a given metric, which should be selected to be correlated to the timing behavior of the monitored task. The metric selection is

architecture and application dependent, and should be performed during profiling activities in the Worst-Case Execution Time (WCET) analysis.
To detect time interference, a set of two rules has been defined:

1. A time interference is detected when the metric value for any given execution is above a detection threshold *D*
2. A time interference is detected when the metric value for α consecutive execution is between a warning threshold *W* and the detection threshold *D.*

Once the metric has been selected, proper thresholds must be selected to implement detection and recovery mechanisms. Threshold selection is performed in a probabilistic fashion, as in the probabilistic WCET (pWCET) analysis approaches. To keep design complexity as low as possible, the complex approaches based on the extreme value theory (EVT) used in the pWCET analysis have been simplified in this technique.
Considering the metric as an aleatory variable X, W and D thresholds are defined as

$$W \in \mathbb{N}^+ : P(X \geq W) = C_1$$
$$D \in \mathbb{N}^+ : P(X \geq D) = C_2$$

where $C_1$and C_2 are two confidence levels. If the metric behaves as a normal distribution of mean μ and variance σ^2, then D and W can be defined as

$$W = \lceil \mu + 2\sigma \rceil$$
$$D = \lceil \mu + 3\sigma \rceil$$

and as consequence, $C_1 \leq Q(2)$ and $C_2 \leq Q(3)$, where $Q(x)$ is the complementary cumulative distribution function for the standard normal distribution.
To explain the area threshold α, it is useful to consider, which shows how the metric values increase when interfering applications are performing continuous accesses to a shared resource, in this case memory, not respecting the constraints provided by the designer. This is a misbehavior, which could be induced by the activation of a software bug. As can be seen in, the distribution of the nominal case and the distributions in the case one or more bugs are active in the system are partially overlapped. This means that if the technique would be based on a simple threshold, its false positive/detected ratio would be highly sensible on the selected confidence level. The area threshold is based on the observation that in the case a bug is active in the system, the probability for a given run to be in the range [W, D] is higher than in the nominal case. Given that the probability in the nominal case of $X \in [W, D]$ is

$$\delta \stackrel{\text{def}}{=} P(W \leq X \leq D) = P(X \geq W) - P(X \geq D) = C_1 - C_2$$

then the probability of α consecutive runs to be in the region [W, D] given that the system is in the nominal case is δ^α. To find a value for α, we put a constraint on δ^α

$$\delta^\alpha \leq C_3 \Rightarrow$$
$$\Rightarrow \alpha = \log_\delta C_3$$

where C_3 is a level of confidence.

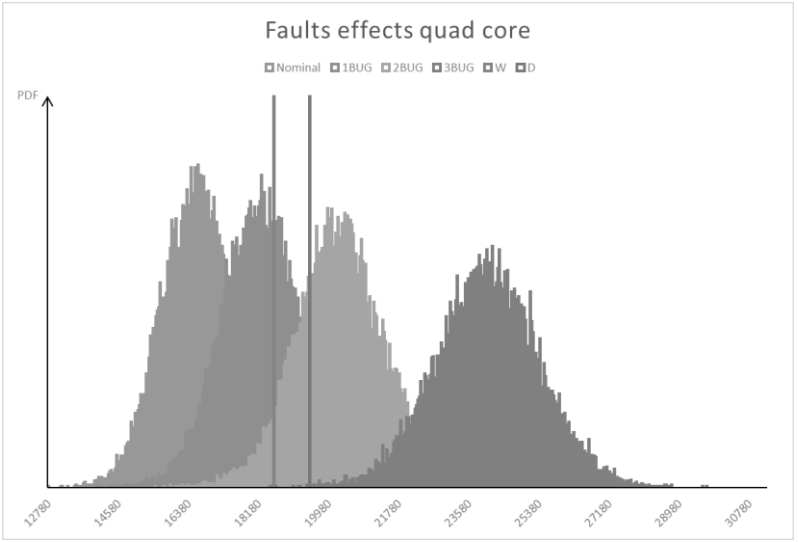


Figure 30 Interference on the metric due to faulty behavior on a quad-core architecture

The detection rules should be implemented in the operating system (OS) as part of the scheduler. Once the system scheduling has been determined using one of the many available techniques for mixed-criticality scheduling on multicore architectures, the proposed technique can be used to control the runtime behavior and make the scheduling reactive to performance issues. Two recovery actions can be implemented:

1) *Graceful degradation:* the scheduler reacts to detection of temporal interference by withholding non-critical tasks from running in parallel with critical tasks. This causes a loss in performance for the non-critical tasks, which should be acceptable for the nature of the non-critical tasks.
2) *Switch to spare computer:* in a configuration where an hot-standby spare computer is available, the reaction to a violation of one of the rules is to switch to the spare computer, while recovering the faulty computer by reset or any other suitable mean.

10.2 Description of the demonstrator

The proposed architecture has been implemented targeting the Zynq System-on-Programmable Chip (SoPC), and in particular two versions of the PoC has been developed:

- Virtual prototype of the PoC: this version leverages the Mentor Graphics VISTA environment to build a virtual prototype of the PoC where an instruction-set simulator is used to model the Cortex A9MP processor embedded in the Zynq device and System C is used to model the watchdog architecture we developed. Using the VISTA simulator features, a set of fault injection scripts have been developed.
- Physical prototype of the PoC: we modeled the watchdog architecture we devised using the VHDL language, and we synthesized the proposed architecture on a ZED board. A laptop is connected to the ZED board via serial cable, as well as via Lauterbach TRACE32 debugger for I/O and fault injection purposes.

Two versions of the proposed software architecture have been developed based on two different virtualization platforms: SysGo PikeOS and Mentor Graphics Nucleus secure.
Two applications stemming from the avionic living lab have been integrated on the proposed hw/sw architecture.

10.3 Conditions for use, downloads

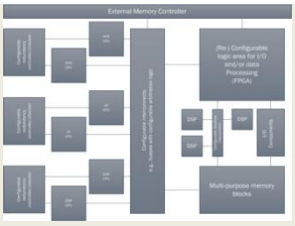
This demonstrator is provided by partner POLITO.

Contact: POLITO, Contact person: Massimo Violante massimo.violante@polito.it
Tel: +39-0110907092

11. MCENoC Demonstrator: A Network-on-Chip for Mixed-Criticality Embedded Systems (18H UoBR)

Technology lanes covered:

- *Heterogeneous Multiprocessor SoC architectures*
- ***Dynamic reconfiguration on HW accelerators and reconfigurable logic***
- ***Networking***
- *Virtualization and* ***verification technologies***



The Mixed Criticality Embedded Network on Chip (MCENoC) is a NoC implementation aimed at dealing with the challenges of providing a compute platform that allows mixed-criticality software to be easily verified and certified. It achieves this by using technologies and techniques that provide tight, statically determinable timing properties, provable through formal methods.

This demonstration platform combines recent work with the switching architecture and verification techniques developed during the project, which have previously been contributed through deliverables D4.4 and D4.12. More recent developments, necessary to create the demonstration platform, include:

- Further engineering effort and formal verification of the switching architecture.
- Selection of a candidate processor, PicoRV32, based on the RISC V architecture.
- Development and formal proof of a network interface for the processor core, via instruction set extension.
- Integration of cores and network onto an FPGA platform – the MACCS development board developed by UoBR, using a Kintex-7 device.
- Software tools for low-level interaction with hardware, providing a route to port software onto the system.

The rest of this section provides further details on these contributions.

11.1 Formal verification of MCENoC implementation

In D4.12 our switch implementation was described, and a presentation was given at the Cadence CDNLive conference within the academic track [20]. Since this, a full paper on the switching architecture and formal verification has been presented at the IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, Lyon, France [21]. This work provides more details on the architecture, its uses, and early performance metrics in terms of throughput, latency and FPGA utilisation.

Following this, we have performed further formal verification, extending into the interface between processors and the MCENoC archtiecture. The network interface is at the bridge between network and system level, which were two previously defined scopes for the definition of requirements.

An example of this is a new system level requirement that states “Attempted interaction with the network must block for no more than a fixed duration.” This ensures that no processing node is permanently halted by, for example, a deadlock scenario that could have arisen in the software. Property definitions in response to this type of requirement relate to the instructions issued by the processor, and so now begin to encompass the software of the system as well, albeit as the lowest level of abstraction.

Additional formal verification was carried out in three parts:

1. With an unconnected network interface, proving general behavioural properties and allowing simple input assumptions to be made without pre-conditions involving the larger structures of the network.

2. In a feedback, loop, where the interface connects to itself, allowing full protocol behavioural testing and data-flow, without the MCENoC.
3. In a fully connected configuration, where the MCENoC is present and so data flows and latencies can be proven.

Further details of this work is planned to be disseminated in a journal publication. This development and verification effort brings the system into a state where it can be integrated and provided as a demonstration platform.

11.2 Integration of the system

The completed first revision of the MCENoC architecture is integrated with a RISC V based, open source processor. RISC V was chosen due to its license, flexibility and the availability of a GCC-based toolchain[5], which allowed for a rapid integration effort.

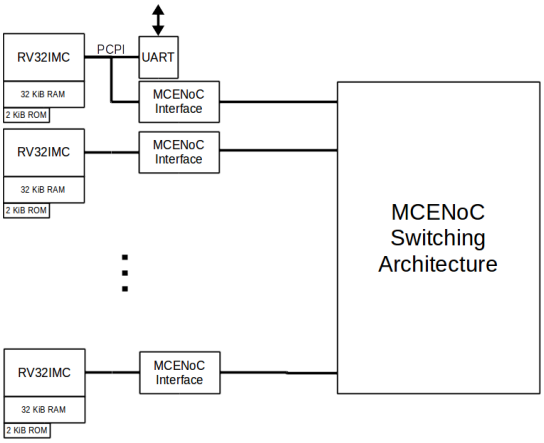


Figure 31: Simplified diagram of MCENoC integrated with RISC V cores

Several RISC V implementations exist from which we selected PicoRV32. This was chosen for its size and relative simplicity, allowing a large number of cores to be synthesised to our target FPGA platform, creating a scale of system appropriate for demonstrating the MCENoC. The cores are configured to support the RV32IMC ISA[6], which includes standard integer operations (I), multiplication extensions (M) as well as compressed ISA (C). To provide time-deterministic execution of code, the barrel shifter option is enabled, otherwise the completion time of shift operations would be variable.

At the system level, each core is given 32KiB of RAM and 2KiB of ROM, provided through the FPGA's Block RAM cells. The memory is configured for single-cycle access, and due to the size, it is beneficial to have compressed ISA enabled. This allows the standard libraries (provided through newlib) to fit in most cases observed during tests so far.

Two processor extensions were developed, through the PicoRV32's PCPI feature, which allows external components to respond to reserved spaces in the ISA. In this case we provide pcpi_uart, which provides instructions for directly accessing an OpenCores UART implementation, and pcpi_mcenoc_ni, which is a network interface into the MCENoC switching architecture. We opted to integrate these into the instruction set, via the "custom0" instruction space, rather than via memory mapping, to avoid any possible contention should the memory hierarchy be modified in the future (for example to use AXI with multiple contentious devices). This preserves the time deterministic nature of each core, which is necessary to fully exploit the deterministic nature of the network. The network interface is formally verified, as described in the previous subsection.

5 https://github.com/riscv/riscv-gnu-toolchain
6 https://riscv.org/specifications/

11.2.1 Hardware platform

The development platform we are using is the MAny Computer Cores System (MACCS), which was designed internally by UoBR. The Trenz Electronic (TE) 0741 System on Module (SoM) is used with this board, which contains a Xilinx Kintex 7 K160T FPGA. MACCS has minimal I/O, but does have a UART and USB, as well as multi-gigabit transceivers for building multi-FPGA projects. Most importantly, the chosen Kintex 7 SoM has sufficient room to accommodate several processor cores and the MCENoC implementation.

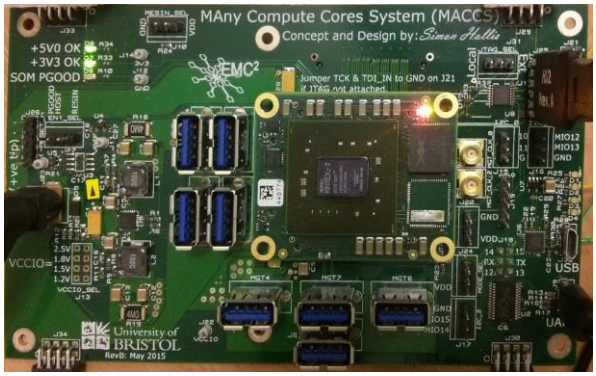


Figure 32: MACCS board with TE0741 SoM

Project partner 18D Sundance uses TE SoMs in its EMC2 development platform[7], which was released in June 2016. There is therefore a relatively simple path towards porting the work delivered by UoBR onto this platform, should it be beneficial to do so.

11.2.2 Synthesis, utilisation & performance

The current implementation of the demonstrator platform features 32 PicoRV32 cores in the previously stated configuration, along with the MCENoC network. The total RAM across the system is 1MiB. The network is configured with an 8-bit datapath and the switching element size (number of ports) is configurable to allow implementation and scaling experiments..

Device utilisation is high in this configuration, primarily due to the number of PicoRV32 cores that are present. 87% of LUTs are used and 84% of BRAMs. DSP blocks are not targeted by the PicoRV32 and there is the potential for modifying the implementation to exploit these, which may improve timing by providing more a more compact core implementation. However, this is out of the scope of current efforts.

The system operates fully synchronously at 100MHz. It is evident from timing searches that the speed limitation is due to delays between BRAMs and processor logic, rather than within the MCENoC network itself, which has previously been synthesised at 4x this speed. The choice of PicoRV32 features, to improve performance per clock and timing predictability, has also influenced the achievable operating frequency.

In this configuration, the bisection bandwidth of the MCENoC is 25.6 Gbps, which given the frequency and data-width difference, compares as expected with the 11.6 Gbps of the 1-bit wide, 32-node, 364MHz version stated in [20].

11.3 Software and tools

To enable software to target the MCENoC platform, several software tools have been developed. These include:

- A simple driver for the UART provided to the PicoRV32 through PCPI instruction extensions. This includes interrupt support.
- A driver for the MCENoC Network Interface (NI).
- A bootloader for the PicoRV32's ROM, allowing programs to be loaded over UART (fully functional) or MCENoC NI (work-in-progress).
- A syscall handler, implementing a subset of the stubs provided through newlib.
- A Python based route calculator for network permutations, providing configuration bits for communication phases between nodes.

The syscall handling provides sufficient functionality to enable standard read/write functions with stdin and stdout directed over the UART interface. This allows programs to be compiled with standard tools and libraries, using a version of GCC that targets RISC V's RV32IMC specification.

[7] http://www.sundance.technology/system-on-modules-som/som-modules/te0715-7030-xilinx-zynq-arm-fpga-som

The route calculator provides the header bits needed for node A to establish communication with node B, in the presence of communication between other nodes, up to all nodes communicating. Provided no destination is sought twice in a single communication phase, the route calculator will provide valid routes for all nodes.

This static routing method is based on Waksman's [22] routing algorithm for Benes networks. It is extended to consider the switching element size at each network stage, which is configurable in an MCENoC implementation. Route setup for MCENoC is performed in-band by the source nodes, whereas Waksman's algorithm assumes a global control plane, so a modification is made to map between these two configuration methods. A routing phase for a 8192 node system can be calculated in 0.1 seconds on an Intel Haswell-based laptop, and scales linearly with the number of nodes.

11.4 Future and continuing work

With a functioning and formally verified implementation of the MCENoC achieved, use case experiments are now possible. We define three main work areas that can potentially be explored.

11.4.1 Enhanced software libraries

To ease the porting of code to the platform, the basic set of drivers and standard libraries can be extended. In particular, the declaration of communication, production of routing data and managing of routing phases at run-time would benefit from tight integration. This requires further work on software libraries to provide a higher level communication interface, integrate the routing calculations at compile time, and handle their enforcement at run-time through a simple kernel or communication manager task.

11.4.2 Performance-oriented evaluation

In order to evaluate the MCENoC against other network configurations, example programs must be selected for comparison. Either the programs can be ported to the platform, or their communication patterns simulated on the platform. Several communication structures are of interest, such as fixed grids and hypercubes, as well as more dynamic structures such as neural networks and large-scale graphs.

11.4.3 Mixed-criticality case study

This focuses on evaluating the timing and route-forming guarantees of the MCENoC in a context where multiple tasks, potentially with very distinct communication patterns, co-exist on the device. This can be achieved with a synthetic combination of tasks from the performance-oriented evaluation, or be based on use cases from other project partners, with particular focus on automotive or avionics related activities. The ultimate goal is to demonstrate the conditions in which high criticality tasks cannot be disrupted by those of a lower criticality.

11.5 Availability and conditions of use

Available to EMC2 partners upon request to steve.kerrison@bristol.ac.uk where access to UoBR's Git repositories for the software and hardware will be given. We plan to make these resources publicly available under license compatible with RISC V in the future.

12.Conclusions

This final report about enhanced demonstration platforms including basic innovative techniques overview should give an summary about our demonstrator activities and can be used as knowledge base for further investigations and projects in future. There are a lot of Hardware description and software links, which can be used also for public.

13.References

[1] John McDougall: Simple AMP: Zynq SoC Cortex-A9 Bare-Metal System with MicroBlaze Processor, XAPP1093 (v1.0.1) January 24, 2014 http://www.xilinx.com/support/documentation/application_notes/xapp1093-amp-bare-metal-microblaze.pdf

[2] Trenz Electronic TE0720-02-2I Series (Z-7020) http://www.trenz-electronic.de/products/fpga-boards/trenz-electronic/TE0720-02-2I-zynq.html

[3] TE0701 Carrier Board User Manual https://wiki.trenz-electronic.de/display/4X5B/TE0701+Carrier+Board+User+Manual

[4] Zynq-7000 All Programmable SoC Technical Reference Manual UG585 (v1.8.1) September 19, 2014 http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

[5] Vivado Web Install Client - 2015.2 Lightweight Installer Download http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2015-2.html

[6] Software Development Kit Standalone WebInstall Client - 2015.2 Lightweight Installer Download http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2015-2.html

[7] PicoBlaze 8-bit Embedded Microcontroller User Guide for Extended Spartan 3 and Virtex5 FPGAs; Introducing PicoBlaze for Spartan-6, Virtex-6, and 7 Series FPGAs. UG129 June 22, 201. http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf

[8] EMC² – 'Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments' is an ARTEMIS Joint Undertaking project in the Innovation Pilot Programme 'Computing platforms for embedded systems' (AIPP5). http://www.artemis-emc2.eu/

[9] 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

[10] "Asymmetric Multiprocessing on ZYNQ with EdkDSP accelerators onXilinx ZC702 board - ISE 14.5"

[11] http://flextiles.eu

[12] http://qt-project.org/wiki/Qt_5.0

[13] AVNET, „HDMI Input/Output FMC Module," 26 08 2015. [Online]. Available: http://www.em.avnet.com/en-us/design/drc/Pages/HDMI-Input-Output-FMC-module.aspx

[14] Sundance, „EMC2-DP – PC/104 ONEBANK CARRIER FOR SOC MODULES," 26 08 2015. [Online]. Available: http://www.sundance.technology/som-cariers/pc104-boards/emc2-dp/

[15] AVNET, „ON Semiconductor Image Sensor with HDMI Input/Output FMC Bundle," 26 08 2015. [Online]. Available: http://www.em.avnet.com/en-us/design/drc/Pages/OnSemi-Image-Sensor-with-HDMI-Input-Output-FMC-bundle.aspx.

[16] Trenz, „Trenz WiKi - Lattice CPLD Programming," 25 08 2015. [Online]. Available: https://wiki.trenz-electronic.de/display/4X5B/TE0701+JTAG+Programming+Guide#TE0701JTAGProgrammingGuide-LatticeCPLDProgrammingLatticeCPLD

[17] El Salloum, C., Elshuber, M., Höftberger, O., Isakovic, H., & Wasicek, A. (2013). The ACROSS MPSoC - A new generation of multi-core processors designed for safety-critical embedded systems. *Microprocess. Microsyst., 37*(0141-9331), 0141-9331.

[18] VMware Workstation 12 Player for Windows 64-bit. https://www.vmware.com/products/player/playerpro-evaluation

[19] SDSoC - 2015.2 Full Product Installation - 2015.2 Full Product Installation http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/sdx-development-environments/sdsoc/2015-2.html

[20] S. Kerrison and K. Eder. “Formal Verification of Predictable Non-Blocking Switches in Multi-Core Mixed-Criticality Systems”. In: Cadence CDNLive EMEA. Munich, Germany, May 2016.

[21] S. Kerrison, D. May, and K. Eder. “A Benes Based NoC Switching Architecture for Mixed Criticality Embedded Systems”. In: IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC 2016). Lyon, France, September 2016, pp. 1–8.

[22] A. Waksman, “A Permutation Network,” Journal of the ACM, vol. 15, pp. 159–163, Jan 1968.

[23] Haris Isakovic, R. G. (2016). A heterogenous time-triggered on a hybrid system-on-a-chip platform. *ISIE 2016*. Santa Clara, CA, USA: IEEE.