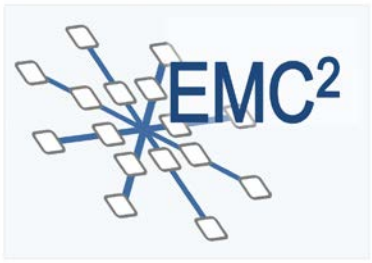


Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments

Project Acronym:

EMC²

Grant agreement no: 621429

Deliverable no. and title	**D12.6 – Final evaluation of the innovation results**	
Work package	WP12	LL Cross Domain Application
Task / Use Case	T12.1 T12.2 T12.3 T12.4 T12.5	UC Seismic surveying UC Video surveillance for critical infrastructure UC Medical imaging UC Control Applications for critical infrastructure UC Railway applications
Subtasks involved		
Lead contractor	Infineon Technologies AG Dr. Werner Weber, mailto: werner.weber@infineon.com	
Deliverable responsible	Fornebu Consulting Hans PetterDahle mailto: hans.petter.dahle@fornebuconsulting.com	
Version number	V1.2	
Date	10/04/2017	
Status	Final	
Dissemination level	Public (PU)	

Authors

Partici-pant no.	Part. short name	Author name	Chapter(s)
12A	Fornebu	Hans PetterDahle	Initial "empty" document. Chapter 2.
02G	AIT	Axel Weissenfeld	Chapter 3 “Passenger tracking”.
12C	Simula	Xing Cai	Chapter 2.
01M	eVision	Michael Geissel	Chapter 3.
17D	Philips	Mark van Helvoort	Chapter 4 and 7 “Medical Imaging”.
17D	Philips	Bastijn Vissers	Chapter 4 “Medical Imaging”.
15Q	ITI	Ismael Salvador	Chapter 2.
04A	BUT	Roman Juranek	Chapter 3.
02J	TAT	Peter Tummetlshammer	Chapter 5.
02J	TAT	Oscar Medina	Chapter 5.
15L	HIB	Raúl Santos d ela Cámara	Chapter 3.
12B	WesternGeco	Bjørn Nordmoen	Chapter 2.
12C	Simula	Stuart Clark	Chapter 2.
16J	ABB	Tiberiu Seceleanu	Chapter 5.
17G	TU Delft	Imran Ashraf	Chapter 4.
12C	Simula	Geir Yngve Paulsen	Chapter 2.
01U	Siemens	Jürgen Salecker	Chapter 3.

Document History

Version	Date	Author name	Reason
v0.0	25/02/2017	Hans PetterDahle	Initial document structure
v0.0	25/02/2017	Hans PetterDahle	Initial text for Ch 1 and Ch 2
v0.1	02/03/2017	Axel Weissenfeld	AIT - Initial text for Ch 3
v0.1	02/03/2017	Xing Cai	Updated Chapter 2, sections 2.2 and 2.4.1
v0.1	02/03/2017	Michael Geissel	Updated Chapter 3
V0.1	09/03/2017	Mark van Helvoort	Added “Medical Imaging” to Chapter 3
V0.1	09/03/2017	Ismael Salvador	Updated Chapter 2
V0.1	13/03/2017	Roman Juranek	Updated Ch 3 (BUT)
V0.1	14/03/2017	Peter Tummetlshammer	Added Railway use case (Chapter 5)
V0.1	15/03/2017	Raúl Santos de la Cámara	Added UC12.2.5 Access Control
V0.1	17/03/2017	Hans PetterDahle	Updated Ch 2
V0.1	17/03/2017	Peter Tummeltshammer	Added Cybersecurity aspect to WP 12.5
V0.1	20/03/2017	Tiberiu Seceleanu	Updated Chapter 5.
V0.1	21/03/2017	Imran Ashraf	Updated Section 4.1.3 and 4.2.3
V0.1	27/03/2017	Peter Tummeltshammer	Added Railway conclusion
V1.0	01/04/2017	Hans Petter Dahle	Incorporated the feedbak from the reviewer
V1.1	04/04/2017	Jürgen Salecker	Added Siemens contribution
V1.2	10/04/2017	Hans Petter Dahle	Updated author list and document history, minor editing of Simens contribution.

Publishable Executive Summary

This deliverable contains evaluations of the final innovation results from the individual use cases in WP12 in the ARTEMIS project EMC². These use cases originate in different domains; the main objective, shared by all of them, is to exploit results from previous research projects and from the EMC² technological work packages and to use them in innovative ways to prepare the ground for future multi-core applications. A description of the intended innovation per use case is included.

These use cases are:

- UC12.1: Seismic surveying by ship
- UC12.2: Video surveillance for critical infrastructure
- UC12.3: Medical imaging
- UC12.4: Control applications for critical infrastructure
- UC12.5: Railway applications

Evaluation of the innovation results are based on previous requirements, specifications, designs and experience for the WP12 prototypes (described in D12.1- D12.5, see References [1] – [5]). Since D12.2 - D12.5 are Confidential, and D12.6 is Public, some parts of D12.2 - D12.5 are repeated in D12.6, when needed.

The evaluations show that all the WP12 tasks have achieved significant innovation results. Time, efforts and resources needed to develop and execute the multi-core applications of the use cases have been significantly reduced.

Table of contents

List of figures

List of tables

1. Introduction

During the EMC2 project two of the WP12 partners had to leave the project. These partners were both tool vendors, and they co-operated with other WP12 partners in Task 3 on the medical use case and in Task 4 on the tool integration. It was not feasible to replace these two tool vendors, and the ambitions of Task 3 and Task 4 had to be adjusted, taking into account that some activities could not be executed as planned.

1.1 Objective and scope of the document

The purpose of this document is to evaluate the final innovation results from the individual use cases in WP12 in the ARTEMIS project EMC². These use cases originate in different domains; the main objective, shared by all of them, is to exploit results from previous research projects and from the EMC² technological work packages and to use them in innovative ways to prepare the ground for future multi-core applications. A description of the intended innovation per use case is included.

The use cases' prototypes adhere to requirements and specifications of D12.1 and follows the design described in D12.2. The initial prototypes were described in D12.3, while D12.4 contained the results of evaluating the initial prototypes. D12.5 described the final prototypes and the intended innovation.

In WP12 (Cross domain applications), five use cases from different domains are considered:

- UC12.1: Seismic surveying by ship
- UC12.2: Video surveillance for critical infrastructure
- UC12.3: Medical imaging
- UC12.4: Control applications for critical infrastructure
- UC12.5: Railway applications

1.2 Structure of the deliverable report

The document has five main chapters, one for each use case. Each of these chapters contains the following sections:

- Introduction to the use case and its evaluation criteria
- Overview of the use case implementation, its limitations, and achieved features
- The intended innovation
- Evaluation results

The last section contains two subsections. The first subsection is an evaluation of the outcomes from the use case. The second subsection assesses current and potential impact of the innovation results.

In addition to the five main chapters, there is first a chapter with an introduction (which you are reading now), and a final chapter with conclusions.

In this document we are referring to the requirements of D12.1[1]. The format of these references are UC12.n-R00nn, where ‘n’ is a digit. Below you see an example text that illustrates how requirements are referred to:

Two architectural possibilities have been investigated to fulfil requirement UC12.3-R0060.

2. UC12.1 – Seismic surveying

D12.1 [1] contains a description of the seismic use case and its objectives. It describes the current way of working at WesternGeco when developing software for multi-core computations, and it describes ideas for a new approach of automating parts of the manual software development work. This new approach has been implemented in Use Case 12.1. The requirements to the use case are documented in D12.1, which also describes quantitative evaluation metrics to be used in the assessment of its results.

2.1 Introduction to the use case and its evaluation criteria

Even though the use case is described in full in D12.1, we give a brief refresh of it and its evaluation criteria in this section.

To realize the ultimate goal of real-time processing, on board a seismic survey vessel, where 3D seismic data is continuously collected, the numerical algorithm adopted must be able to release the full potential of the on-board hardware. In this strive, engineers at WesternGeco must test numerous numerical strategies, while also having to efficiently implement these on modern multi-core hardware.

MATLAB® is a popular software for computational mathematics, particularly because of its accessibility for scientists and engineers as a high-level scripting language. This ease of use, and a large library of toolboxes, make MATLAB a good choice for testing and prototyping new algorithms.

However, once the algorithms are tested, the ability to sufficiently optimise MATLAB code becomes a key concern. From a conceived numerical algorithm to its full-fleshed parallel implementation, the only existing implementation alternative follows a manual approach. More specifically, the engineers start with a blueprint of the numerical algorithm written in MATLAB. Then, the Matlab code is manually translated into a C++ code using the Armadillo library, which has a syntax close to MATLAB. Thereafter, additional manual programming effort is invested to parallelize and optimize the C++ code.

Task 12.1 aimed to inject automated code translation and optimization, as much as possible, into the above described implementation cycle. This is a two-stage process. First, an input MATLAB code is automatically translated into an equivalent form using the Armadillo C++ library. Then, the C++ intermediate code is further transformed into a parallelized and vectorized form.

The table below gives an overview of the evaluation metrics as defined in D12.1.

Time measurements of running the code generator.
Time measurements of running the generated code.
Calculate the cost of using the code generator at project end.

Table 1: Evaluation metrics for the seismic use case

2.2 Overview of the use case implementation, its limitations, and achieved features

In Use Case 12.1 we have developed a source-to-source code translator that automatically converts an input MATLAB code into a C++ code as output. This code translator is called *m2cpp*.

MATLAB and C++ are two fundamentally different languages. Some inherent language features of MATLAB pose challenges to an automated code translator. Two examples are implicit typing of variables and multiple return values from functions. To handle these inherent challenges, existing MATLAB translators ask the user to insert annotations of variable declaration and initialisation. Such an invasive approach is not very user-friendly, and may also cause problems if algorithm developers want to further change the MATLAB code.

The aim of *m2cpp* is to facilitate an automatic and non-invasive translation from MATLAB code – for seismic processing – to the matching Armadillo counterpart in the C++ language. *m2cpp* can translate a subset of the MATLAB language, this subset contains language constructs that are typically used for seismic processing. The translation is non-invasive because the user of *m2cpp* does not need to insert any annotation or extra coding into a serial MATLAB program before passing it to the translator. Automatic identification of MATLAB variable types is done by a pre-processor of *m2cpp*, producing a meta-info file. The *m2cpp* translator itself is written in the Python language, with a tailor made top-down recursive descent parser [7] that follows the same approach adopted by Antlr [6]. The parser reads the input MATLAB code and internally sets up an abstract syntax tree, which is then subjected to a post-order tree walk [7] for necessary code analysis and translation to the resulting C++code. An example of a code analysis task is to identify MATLAB functions that return multiple values, which are translated as additional arguments of the corresponding C++ functions. Another important task of code analysis is about thread-based parallelisation of for-loops, where some variables must be made private per thread to avoid race conditions.

The applicability of *m2cpp* assumes that the input MATLAB code consists of vector and matrix-based computations that are covered by the functionality of the Armadillo C++ library. In addition, *m2cpp* has included a couple of new C++ functions beyond the original functionality of Armadillo, so that typical plotting functions of MATLAB can also be automatically translated. There are two limitations with *m2cpp*, concerning implicit change of variable types and dynamic expansion of vectors/matrices inside MATLAB. We remark that dynamically changing a variable type violates C++'s principle of static typing, whereas dynamic vector/matrix expansion is performance unfriendly and seldom used in MATLAB implementations of engineering computations.

Regarding code parallelization that suits shared memory on multi-core architectures, *m2cpp* is able to automatically insert OpenMP directives or additional code lines following the syntax of TBB, so that iterations of designated for-loops can be divided and executed by multiple threads. The restriction to loop-level parallelism (thus excluding task-level parallelism) is justified by the fact that the heaviest computations almost exclusively happen in for-loops.

2.3 The intended innovation

In WesternGeco/Schlumberger, researchers that develop new algorithms most often use MATLAB for their work. To transform the results of the researchers work into industrialized and commercial products is a major effort executed by our engineering centers. The main goal of our EMC2 activities is to simplify and accelerate this process, by automating the translation process from MATLAB to C++ and at the same time generate code that is multi-core aware. This gives the additional benefit that the code also run much faster. Since performance is so important for us, we handle this as a white-box approach, i.e. we see the need for further manual optimization after the initial generation. Therefore, we decided to use Armadillo C++ template library as a platform for the generated code. Since Armadillo syntax is quite close to MATLAB, it will ease the understanding of the generated code, and therefore make it more easy to continue manual optimization.

In the EMC2 project's Document of Work (DoW) we stated our vision for the results as follows:
Software engineering for multi-cores is a major bottleneck in embedded systems engineering, and T12.1 will make it possible to produce software in less calendar time and at significantly reduced costs. For the development work and embedded systems targeted in T12.1, the aim is to reduce engineering calendar time to 1/10th of what it takes today, and to reduce execution time to 1/5th of that of today's systems.

2.4 Evaluation results

We have used a public domain package called SeismicLab[1] for testing and verification. SeismicLab is a MATLAB seismic data processing package. The package can be used to process small seismic data sets,

[1] http://seismic-lab.physics.ualberta.ca/

and it is mainly intended for research and teaching purposes. We chose this package because it covers a subset of MATLAB that is highly relevant for seismic surveying, and because it is open source, meaning that anyone can look at the code if they are interested.

Out of the 9 demos of the SeismicLab package, *m2cpp* is able to automatically and correctly translate eight of them. The last demo, *spiking_decon_demo.m*, uses MATLAB functions *xcorr*, *levinson* and *hamming*. These functions are not part of Armadillo's functionality, and it would have required too much effort to provide C++ code for them. Therefore *spiking_decon_demo.m* was not executed. *hamming* is also used in the demos *parabolic_moveout_demo.m* and *va_demo.m*. WesternGeco provided a C++ implementation for *hamming*, so that these two demos could be executed as part of the evaluation.

Our evaluation is based on the measurements and experiments gained from eight SeismicLab demos. Thus, when we talk about the MATLAB code in the following parts of this document, we mean the MATLAB code of the 8 demos and the files they include.

2.4.1 Evaluation of outcomes from the use case

In addition to run-time performance improvements, a key goal has been to radically reduce development time for transforming research prototypes in MATLAB to production code in C++. In the UC12.1 experiments we have experienced a dramatic reduction of development time. We estimate that developing with *m2cpp* is 10 to 100 times faster, compared to the old way of working. I.e. the "new development" time is from 1% to 10% of the "old development" time. This goes both for calendar time and person hours used. The variation from 1% to 10% is due to the following. For some cases the MATLAB code contains functions which are not supported by Armadillo C++, i.e. there exist no counterpart in Armadillo to the MATLAB funtion. Then the function must be coded manually. We have over time developed our own library with such functions, as we need them.

For all the eight demos, the auto-translated C++ code by *m2cpp* runs considerably faster than the original MATLAB counterpart. The largest performance gain arises from computations that are implemented by plain vector and matrix operations (i.e. without invoking functions that are available from special math libraries such as Intel's MKL library). As an example, the following table shows the wall-clock time usage of the *parabolic_moveout* demo, run on a server with dual-socket Sandy Bridge CPUs (of model E5-2670). We thus compare the performance of the original single-core MATLAB code against the auto-translated C++ code which is compiled by `icpc –Ofast –xHost –DNDEBUG –DARMA_NO_DEBUG`. The input data file size has been enlarged to 4004x1568 by interpolating the original input data file.

	MATLAB	**C++**	
Serial performance	261.7 s	59.5 s	
Parallel performance	*N/A*	**OpenMP**	**TBB**
2 threads		30.1 s	30.0 s
4 threads		15.5 s	15.3 s
8 threads		8.1 s	7.9 s
16 threads		4.4 s	4.4 s

Table 2: Evaluation results for the seismic use case

The single-core performance of the auto-translated C++ version of parabolic_*moveout* is 4.4x faster than the serial MATLAB counterpart. The auto-parallelization of *m2cpp* (for both OpenMP and TBB) increases the performance improvement factor to 59.4x by using 16 threads. It should be noted that MATLAB's *parfor* construct refuses to parallelize the main for-loop even though its iterations are independent, which is utilized by the *m2cpp* translator.

Apart from adopting a non-invasive and thus user-friendly approach, another strength of *m2cpp* is that the auto-translated C++ code retains the same coding structure and variable names as the original MATLAB code. This is illustrated by the following sample code:

Original MATLAB:

```
for it = 1:ntau
  for iq = 1:nq
    time = tau(it) + q(iq) *
           (h/hmax).^2 ;
    s = zeros(2*L+1,nh);

    for ig = -L:L;
      ts = time + (ig-1)*dt;

      for ih = 1:nh
        is = ts(ih)/dt+1;
        i1 = floor(is);
        i2 = i1 + 1;

        if i1>=1 & i2<=nt ;
          a = is-i1;
          s(ig+L+1,ih) =
              (1.-a) * d(i1,ih)
              + a * d(i2,ih);
        end;
      end
    end

    s = s.*H;
    s1  = sum( (sum(s,2)).^2);
    s2  = sum( sum(s.^2));
    S(it,iq) = s1-s2;
  end
end
```

Auto-translated C++:

```
for (it=1; it<=ntau; it++) {
  for (iq=1; iq<=nq; iq++) {
    time = tau(it-1)+ q(iq-1) *
           arma::square(h/hmax) ;
    s = arma::zeros<mat>(2*L+1, nh);

    for (ig=-(L); ig<=L; ig++) {
      ts = time+(ig-1)*dt ;
      for (ih=1; ih<=nh; ih++) {
        is = ts(ih-1)/dt+1 ;
        i1 = std::floor(is) ;
        i2 = i1+1 ;

        if (i1>=1&&i2<=nt) {
          a = is-i1 ;
          s(ig+L,ih-1) =
              (1.-a) * d(i1-1, ih-1)
              + a * d(i2-1, ih-1) ;
        }
      }
    }
    s = s%H ;
    s1 = arma::as_scalar( arma::sum(
    arma::square(arma::sum(s,1))));
    s2 = arma::sum(arma::sum(
              arma::square(s))) ;
    S(it-1, iq-1) = s1-s2 ;
  }
}
```

This results in full traceability of every algorithmic pattern and detail, allowing experienced users to carry out manual code optimizations. For the example of *parabolic_moveout*, the serial time usage of a further hand-optimized C++ version is 10.8s, where the parallel time usage is reduced to 1.03s using 16 threads. In other word, the performance improvement factor is 254.1x by using 16 threads and manual code optimizations.

The time it takes for *m2cpp* to automatically translate MATLAB to C++ was also measured. Running the code generator on the Seismic Labs examples are almost instantaneous (time < 1 second).

2.4.2 Assessment of impact

The *m2cpp* tool grew out of the work-process developed within the global HPC-group in Schlumberger. This group had developed a manual process for translating MATLAB to Armadillo C++. *m2cpp* is a tool that automates this process, and it will be very useful in the future work for this group.

Since the entrance treshold for converting MATLAB code to efficient C++ with *m2cpp* is quite low, we expect the tool to be adopted also by several other engineering groups in Schlumberger. For Schlumberger it is important to reduce the time it takes to bring research code into production environments, and there is currently considerable management attention on new ways of working to achieve this.

The *m2cpp* code-generator will be announced and presented to the developer community within Schlumberger at a company-global web-cast in the second quarter of 2017. The code is also available as an open-source project and can be downloaded from https://github.com/emc2norway/m2cpp.

It was and is expected that the T12.1 results can be extended and (re-)used in many similar contexts. The challenges solved in T12.1 are common to all domains where real-time image processing is a key compo-

nent. We expect that the major reductions in development time and execution time to be a key enabler for new and improved multi-core applications.

The same need, for translating high-level MATLAB code into a more hardware-efficient C/C++ version, has been identified for two different oil and gas industry clients by Kalkulo AS, a subsidiary of EMC2 partner Simula Research Laboratory. These clients use MATLAB for prototyping software in several contexts, from stratigraphic modelling to seismic processing and analysis. Kalkulo is later called upon to write professional software tools based on these prototypes. Kalkulo evaluated the usefulness of *m2cpp* for these projects and found that it is especially useful to translate core time-consuming subroutines of the MATLAB software into readable and optimised C++/Armadillo.

3. UC12.2 – Video surveillance for critical infrastructure

Subtask 12.2. combines the activity from multiple companies and research institutes in the field of image processing. The common goal was to create systems that can identify objects or faces with short latency, fast development cycles and affordable system cost in industrial or non consumer applications.

3.1 Introduction to the use case and its evaluation criteria

Consumer applications are developed in huge quantities and manufacturers of those systems can use technologies like Application Specific Circuits that are not affordable for markets that require highly specialised solutions. Those solutions can't afford specific Chip developments and need to build systems with standard components like processors or FPGA devices.

The UC12.2 partners were working very closely together to achieve results in different domains. Those domains were:

- Working with different HW platforms
- Comparing different Algorithms
- Using different Methodologies

eVision

The main objectives for eVision were to identify the best methodology and to find a solution that will enable the company to build smart camera systems that are cost effective and customizable to match different market needs. This requires affordable cost structure and short implementation cycles.

Siemens

The main objective of Siemens was to analyse the various software algorithms and provide recommendations about possible algorithmic and data flow changes. A main goal for Siemens was to enable software engineers to provide/generate professional code (VHDL/Verilog) for FPGAs.

AIT

The table below gives an overview of the evaluation metrics as defined in D12.1.

Req1: Automatic reconfiguration.
Req2: Self-awareness.
Req3: Configuration of QoS manager.
Req4: Data exchange between multiple tracking tasks

Table 3: Evaluation metrics for the AIT use case

ITI

In deliverable D12.4, and according to UC12.2.R0031, ITI used a subset of images from LFW database as a test set. The following measures were considered: True Positive Rate (TPR), False Positive Rate (FPR) and Harmonic Mean (F1), which is a combined measure of the previous ones.

Actions have been taken last months to reduce memory usage and computing time (related to throughput, response time, and discard ratio KPI goals), while preserving detection quality (F1).

HIB

The prototype developed by HI Iberia in WP12 intended to demonstrate the integration of a number of tools and strategies developed in WP2, WP3 and WP5 in an EMC2 application. Tools for variability modelling, performance evaluation and run-time adaptivity, such as PLUM, VIPPE and pappadapt, were integrated into the design workflow of the prototype. The testing was mostly geared towards measuring resource optimization in terms of energy. The prototype simulates a system which is battery duration sensitive, and adaptivity is used to improve overall system runtime.

3.2 Overview of the use case implementation, its limitations, and achieved features

eVision
Together with Siemens and the University of Braunschweig we compared the performance of different algorithms for face recognition and the possibilities to accelerate those algorithms or implement them in FPGA devices that allow HW acceleration.

Object recognition can be based on different architectures:

- Algorithm is executed as software on an external PC/Workstation.
- Algorithm is executed on an embedded controller solution that could be built into a smart camera.
- Algorithm is implemented in a pure FPGA fabric based design that could be built into a smart camera.
- Algorithm is implemented on a SystemOnFPGA, that combines the usage of FPGA logic elements with embedded processor cores (HW or SW cores).
- Algorithm is implemented in an Application Specific Integrated Circuit. This options was not an option for the project since the one-time costs were too high.
- We did not consider the option that the algorithms could be executed on cloud based platforms, since object recognition systems may not be connected to any network.

For Smart Camera applications, the goal was to find an affordable hardware architecture that is flexible enough to implement algorithms and able to do the standard video processing, transporting video data from a source camera to a destination like a monitor or a recorder.

The different use case demonstrators were:

- Algorithm runs on PC (initial demonstrator).
- Simple algorithm (grey conversion) runs on FPGA.
- Object recognition runs as OpenCV implementation on an embedded processor within a FPGA.
- Algorithm for object (face) recognition runs on a SystemOnFPGA, where the algorithmic part is executed on the FPGA logic, and the memory control is handled by the embedded controller.

It turned out that by optimising the algorithms for a later hardware implementation (reduce memory consumption etc.) the system latency on the initial PC implementation could be reduced dramatically (more than a factor 10). Nevertheless, the real limitation of the PC based implementation, is the fact that it is not useable for a smart camera application.

The implementation on a FPGA, without embedded processor, did require a huge effort to create a custom memory controller that is dependent on the structure of the algorithms and specific for the selected target hardware. The real challenge are algorithms that are based on data streams. Hardware acceleration requires parallel processing of the incoming data, and it requires memory to store the video data. Small and affordable devices currently don't have enough memory to store complete HD frames. For that reason, the algorithms need to be rewritten in a way that they are implementable in those chips. That ends up in a compromise of accuracy and supported video resolution. The biggest disadvantage is the long implementation cycle, since the whole design needs be we rewritten in an HDL language. This is so because most of the small devices are not supported by HLS (High Level Synthesis) tools. (The demo board for the pure FPGA design was based on the VSP150 video board with a Lattice ECP3-150 FPGA).

The fastest implementation was achieved by running the algorithm on the ALDEC TySOM platform with embedded Linux and a OpenCV library. That worked very well for testing algorithms, but due to the OpenCV library and the operating systems, the latency was relative high.

The fasted execution was achieved by using the HLS design flow from Xilinx, and by implementing the algorithms on a FPGA with embedded processor without operating system. This was tested on the FPGA

platform that was used and defined by WP4. The same methodology can be used with other FPGA based systems like ALDEC TySOM.

Siemens
The whole algorithm analysis and the change requests were made in very close cooperation with eVision as described already above. New requirements concerning the FPGA platform were discussed, and according to this the algorithms were analysed and dynamically evaluated (maximum integer values, errors induced due to shift from float to integer, etc.), and if possible adapted. At the end the platform was based on a ZYNQ FPGA (fom Sundance, WP4) where the algorithm is "*compiled*" with the help of HLS (High Level Sythensis) onto the FPGA. Impressive performance improvements were achieved, a factor of around 60 if compared with a i5 2,8GHz Laptop.

AIT
We developed a novel middleware for computer vision tasks (exemplarily presented on passenger tracking). It consists of multiple components embedded into individual modules, that follow the micro-services architectural style. Each module acts as an autonomous web service communicating via a Representational State Transfer (REST) interface, which enables distributed processing, so that vision tasks can be split over various units. The middleware is independent of a specific programming language or development environment. It offers a flexible fundament to be integrated into existing or new real-world systems to solve complex computer vision tasks in different application domains. Each module is able to collect data from multiple sources (e.g. visual sensors), processes the data according to its configuration, stores the results for later retrieval, and provides them to multiple consumers. For dividing complex computer vision applications into small tasks (modules), the framework provides the basic source code skeleton of the module with dedicated storage – both for configuration parameters and result data. Each module comes equipped with a monitor of its own performance. In this way, complex computer vision algorithms can be easily split into smaller parts, which are easier to handle. The key module, which manages the reconfiguration and triggering of the execution, is denoted as dynamic input selector (DIS). This module consists of a control and QoS unit, which can be configured by the user.The control unit triggers a reconfiguration whenever there is a resource variation with impact, which implies to select another sub-chain. These sub-chains are pre-configured, so that a system reconfiguration is achieved without significant delay. Based on quality of service parameters provided by the modules, such as the processing duration per frame, the dynamic input selector chooses to operate between multiple sub-chains. The decision is made by analysing the QoS parameters provided by the preceding modules. A hysteresis function prevents the system from self-oscillating between sub-chains.

ITI
The main problems with the initial algorithm were memory and CPU usage. In the final prototype, both have been optimised. In a previous deliverable (D12.5), we showed that by means of a multi-scale dynamic buffer, it is possible to reduce the memory usage. But there was still another problem when using an FPGA, because of the existence of recursive functions. This issue has been solved by converting them to iterative. In particular, the evaluate_buffer function has been modified:

Recursive version:
```
if (!buffer || (buffer->current_row + 1) < buffer->height ||
        depth <= 0) return; // end of recursion
INSTRUCTIONS
buffers_evaluate(buffer->next, depth-1, instruct, table, threshold, faces);
```

Iterative version:
```
while (!((!buffer || (buffer->current_row + 1) < buffer->height || depth <= 0))){
    INSTRUCTIONS
}
```

One dynamic buffer drawback is the following. Due to space restrictions, the sampling step is higher than desired, and some scales are not processed, causing faces in those scales to be lost. This behaviour has been improved, by using virtual scales that allow us to sample the image at a different scale. For example,

if memory buffer is sampled at 1.6, it will search for faces in scales 1.6, 2.6, 4.1, ..., 1.6^n, while virtual step 1.1 will search for faces in scales 1.1, 1.2, 1.3, ... 1.1^n. The performance of virtual sampling at 1.6 is similar to sample at 1.4 real scale, and saves 33% of memory space. The following image shows the aspect of the buffer as scale grows.

Figure 1: Virtual sampling

BUT

The objective is to implement HDR image capture and object detection in FPGA, in particularly on Xilinx Zynq platform (composed from ARM CPU and FPGA). The algorithms used typically for object detection in embedded platforms are Cascade of Boosted Classifiers (e.g. Haar Cascade or LBP Cascade in OpenCV library), and Soft Cascade (WaldBoost) which is typically faster solution. The detector, as a source of information for classification, uses image features. Typical choices are Haar features, Local Binary Patterns (LBP) or various features based on local ranks (LRD, LRP). In this use case, we focused on the WaldBoost algorithm with LBP or LRD features.

Object detection is computationally a very demanding task, and thus cannot be implemented only on processor based embedded system. Using of some accelerator unit is necessary, for example FPGA or GPU unit. The processor part of such embedded system should handle the configuration and communication tasks, and the coprocessor unit will be dedicated only for object detection algorithm. The advantages of hybrid embedded architectures with FPGA are high computational performance, low power consumption, and wide pallet of available peripheral interfaces. The disadvantages for specific tasks could be relatively small memory throughput, or difficult implementation of certain arithmetic operations, such as division or operations on floating point arithmetic.

Figure 2: The camera prototype

The final prototype is a camera (specifically FPGA and Software solution) able to capture video with high dynamic range - 16 bits per pixel. The resulting HDR image is composed from two frames, each of which is captured with different exposure (requirement UC12.2-R011). This HDR camera is currently able to capture a video stream of HD resolution at 60 frames per second (but it can be configured up to Full HD resolutions).

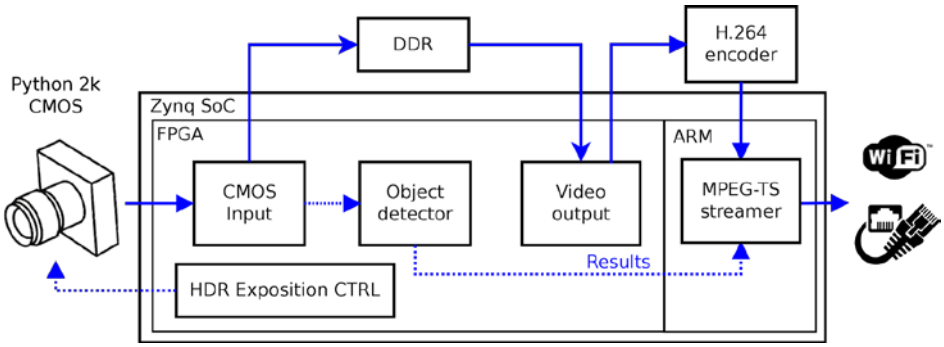


Figure 3: Block scheme of the camera

Figure 3 shows a schematic view of the camera. HDR exposure CTRL controls the CMOS exposure for each frame implemented in the camera. CMOS resolution is 1280x720 pixels at 60 frames per second. We capture images in pairs – one with automatic exposure and one with +2EV (4x exposure time of the first one) – forming HDR image pair. I.e. sensor captures 60 fps, but the output of the camera is 30 fps as two images are captured for each output HDR frame, see Figure 4.On each captured image, the object detection algorithm is executed. The detection results for HDR image pair are merged and post processed (non-maxima suppression to keep only strong detections) on the ARM core, see Figure 4.

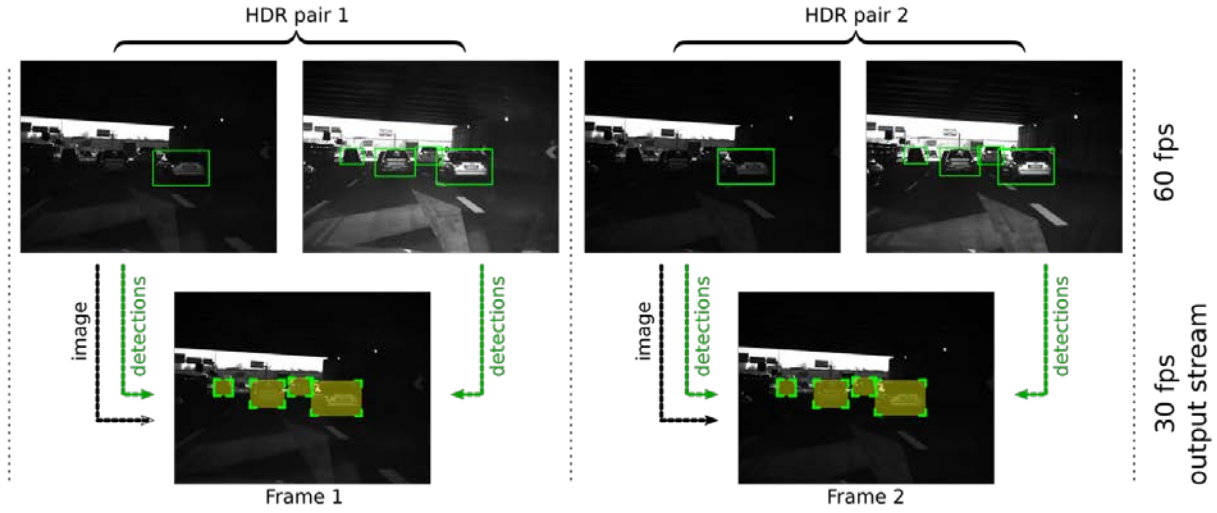


Figure 4: Illustration of object detection process in HDR image pairs.
Top: In each frame objects are detected. Bottom: Detections for each pair are merged and used for output.

The class of detected objects is defined by a microcode of the detector block and it can be changed anytime. The object detector is able to detect objects of various sizes by downscaling the input image. Unlike other state of the art FPGA implementations, we perform image downscaling on-the-fly in FPGA BRAM in order to save the bandwidth to DDR memory. The only limitation is the object size (scanning window size), which is limited to some maximal size during the FPGA synthesis process.

During the project, we considerably improved performance of the object detection algorithm on FPGA and decreased the overall power consumption. We moved to a more powerful, Xilinx Zynq based platform with more resources, which allow us to handle much higher image resolutions than the first prototype. We implemented HDR capture algorithm, which increases robustness of object detection in hard lighting conditions. The output stream of the camera is encoded to 8Mbit H.264 stream (by Fujitsu MB86M25 encoder) and sent via network to clients (serving as IP camera).

HIB

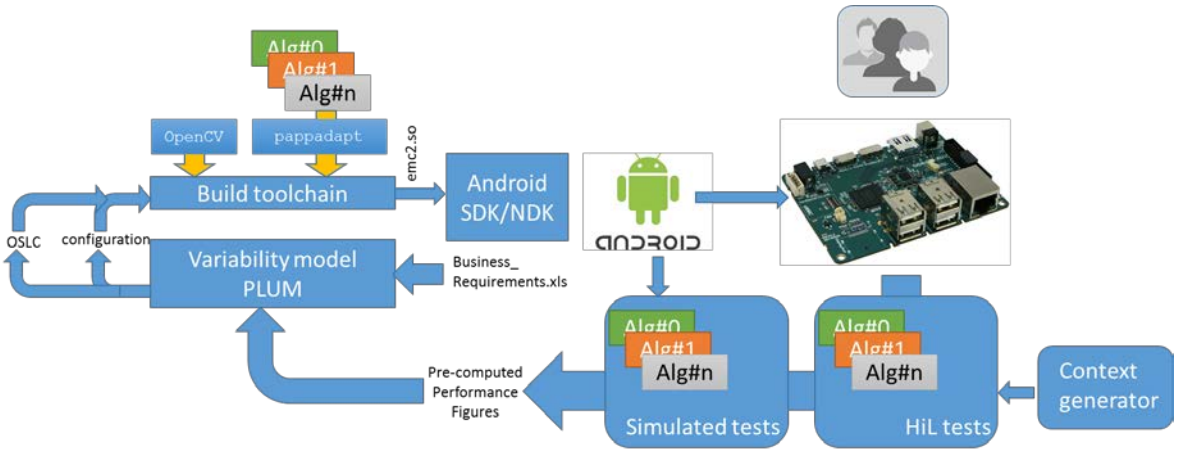


Figure 5: Access Control design

The proposed process of building, testing and reuse of the performance results in the run-time is depicted in Figure 5 above. The process of design starts by providing business requirements, collected in the example using a simple Excel sheet. The variability model in PLUM interprets these requirements and the available assets (here represented by different algorithms in coloured boxes). It provides a configuration, which is transmitted to the build chain using a raw configuration file or through an OSLC bridge. The build toolchain takes the code for all of the algorithms and piggybacks them into a single output file, that, based on the results of run-time evaluation, are chosen by the *pappadapt* library. This is built into an Android application, that is then evaluated for performance and energy expenditure. In the test system, we provide varying user-level QoS requirements using a Context Generation engine. The results of the performance evaluation are fed back into the variability model, so they are compiled into the adaptivity logic (the system knows in advance the estimated performance of the algorithm that will be chosen).

3.3 The intended innovation

eVision

The intended innovation was to find a technology and a methodology that allow building algorithms into smart systems for non-consumer markets. Such systems usually require expensive technology with high development cost.

Siemens

The integed innovation was to provide Software engineers professional support for very high performance computing on embedded devices with limited power budgets. In 4/2016 a project proposal (in a different area, no camera) was defined and presented to the Siemens board (Mr. Rußwurm, etc.) within the newly funded Siemens Innovation fund, however this proposal was rejected as other ones where already more mature.

AIT

The developed middleware for passenger tracking systems tackles the challenges of passenger monitoring on airports where the large extent and the dynamics of the environment, such as the largely variable number of passengers over time, pose significant problems to the design of a surveillance system and its QoS management, especially when hardware and software resources are limited. The proposed software framework solves the problems by

1) its anticipation of the environment and appropriate reaction to the current situation (Req2),
2) a QoS manager configured by the user which carries out a graceful degradation in overload situations (Req1, Req3),
3) subdivision of complex computer vision tasks into multiple simpler tasks, and
4) tasks can be configured individually and executed on different computational nodes.

The data exchange between subtasks is carried out in real-time on the same hardware (Req4), and with a certain delay depending on the network connection. Given a specific QoS to achieve, the presented middleware is able to detect bottlenecks during the execution of a processing chain and to reconfigure it on the fly. Thereby, the reconfigurations are based on a graceful degradation to sustain a QoS and to prevent system failures.

ITI
The developed algorithm allows face detection in architectures with little memory, such as FPGAs, where it is imposible to keep the whole image in memory at the same time. Other code improvements (multi-threading, vectorization, etc, …) can also benefit other platforms such as multi-core devices like cellphones or even desktop PCs.

BUT
During the project, we moved from an initial prototype, which was merely an image pre-processing unit, to a complex IP camera which can be considered a final product with applications in surveillance or industrial quality control. The innovations include higher data throughput of detector IP core, targeting applications which require higher image resolution. We also improved the detection accuracy in hard light conditions (reflections, shadows, etc.) by implementation of pseudo-HDR mode - detection on sequence of differently exposed images.

The target Zynq platform was chosen with regard to easy adaptation of camera for different tasks. Combination of FPGA – with on-the-fly reprogrammable accelerator – and dual core ARM processor – operating full Linux distribution – offers numerous options for easily adapting camera behavior according to customer requirements. All the configuration capabilities of camera parts (image capture, detection, applications in ARM) allow an easy customisation for target applications, and thus faster time to market.

HIB
The usage of performance-aware adaptivity layers that allow the system to choose algorithm implementations knowing in advance how they will affect performance are a novel result. This results in a longer battery life (better energy performance) with minimal penalties on the QoS set by the user.

3.4 Evaluation results

3.4.1 Evaluation of outcomes from the use case

eVision
The optimized algorithms from University of Brno could be used to implement face recognition algorithms into an affordable FPGA device that is on the HW platform from WP4. The used HLS design flow also allows to implement changes very fast.

Siemens
Sucessful algorithm “implementation” on a FPGA device provided by the board from Sundance (WP4), where Siemens bought two setups, in order to have a showcase also internally.

AIT

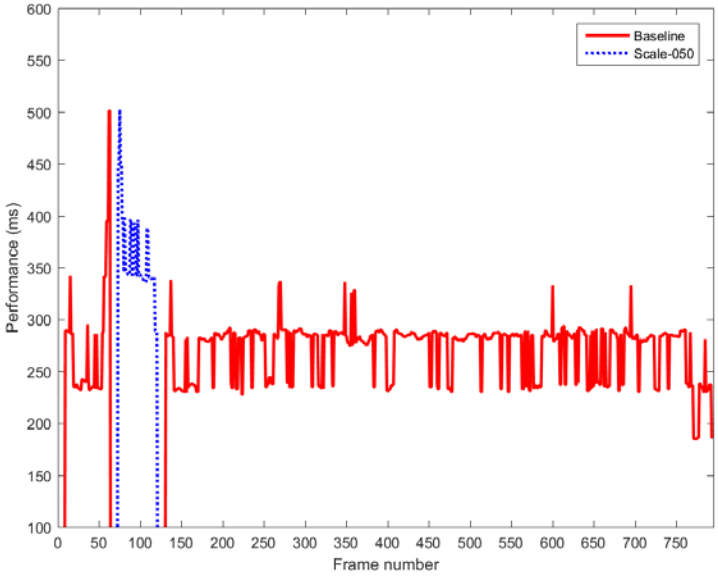


Figure 6: Algorithm performance during run-time of the baseline configuration (red solid line) and the 'Scale-050' configuration (blue dotted line) with the new middleware.

Apart from the quantitative analysis of the object detection algorithm, graceful degradation of system performance controlled by the new middleware was analysed, by measuring the computation time. Figure 6 shows the computation performance of the dynamic switching between two algorithmic configurations, "Baseline" and "Scale-050". The system starts executing the "Baseline" configuration. Around frame 70, the moving average computation time degrades, reaching values between 400 ms and 500 ms. After detecting this deterioration, the system switches to the "Scale-050" configuration which decreases the load of the system. After a certain period of time, the framework detects that the system is not overloaded anymore (around frame 120) and switches back to the initial configuration. During the rest of the sequence, the computing time remains below 350 ms, therefore there is no more need for switching between these two configurations. This switching process assures a certain QoS with sufficient computational resources.

ITI

An improved algorithm that performs face detections in low memory scenarios has been developed and keeps detection quality even if the image data (from the dynamic buffer) is limited. Virtual scales and real scales can be easy adjusted to the final scenario. The code has specially been written to be easily integrated on FPGAs.

The following table shows F1 measure. It confirms that it is possible to improve results when using a virtual scale.

Sampling step	F1	F1 virtual
1.2	81.02	81.26
1.4	77.46	79.49
1.6	74.38	77.23
1.8	67.07	75.33
2.0	60.49	68.11
2.5	46.52	59.01
3.0	42.75	59.96
4.0	29.93	31.52

Table 4: Improved results when using a virtual scale

BUT
Whole camera demonstrator consumes 8W at full load, which includes CMOS at 60 FPS (~1,5W), LP detection on the same speed, H.264 encoding (~1W), and streaming of the video over network. Our implementation outperforms current state-of-the-art implementations in terms of overall performance per pixel, as shown in Table 5. The main benefit is that our architecture is capable of on-the-fly image downscaling, without need of external memory. We are also focusing on low FPGA resource consumption and high detection accuracy, summarized in Figure 7.

	Algorithm	Features	Resolution	FPGA	fps	BRAM	LUT	REG
Cho09	Viola-Jones	Haar	640x480	Virtex5	7	41	67K	22K
Kyrkou11	Viola-Jones	Haar	320x240	Virtex2	64	24	26K	24K
Zemcik2013	WaldBoost	LRD	640x480	Spartan6	160	31	7K	2K
Kyrkou16	SVM	LBP	800x640	Virtex6	40	256	35K	20K
EMC2 Arch.	**WaldBoost**	**LRD**	**1280x720**	**Zynq**	**66**	**38**	**10K**	**7K**

Table 5: Comparison of resource consumption of state-of-the-art FPGA architectures for object detection

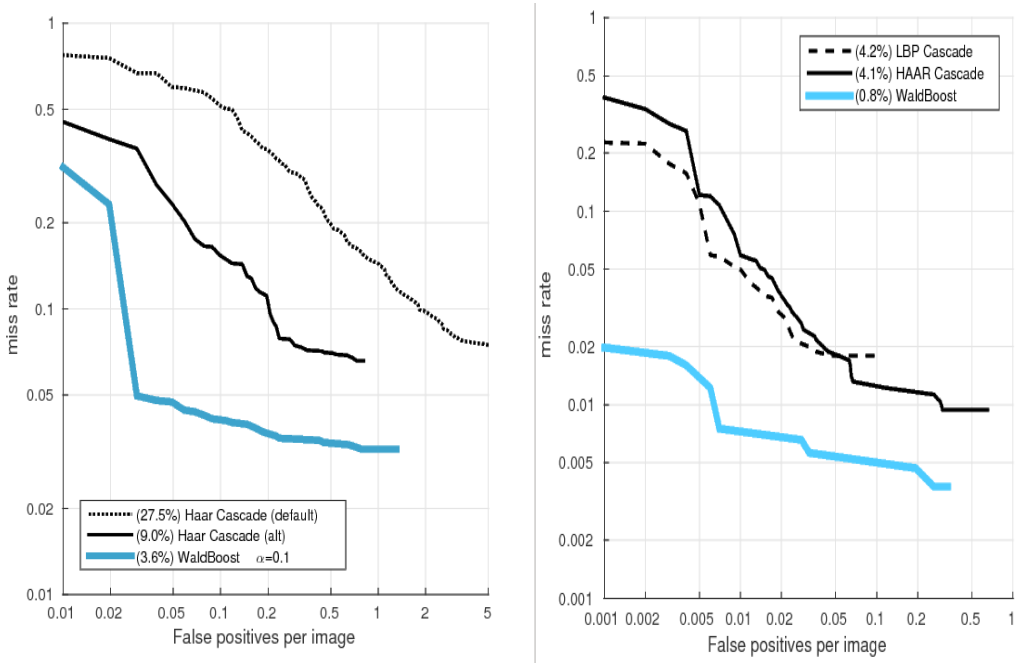


Figure 7: Evaluation of our face detector (left) and license plate detector (right) and comparison to Haar Cascades used in state of the art FPGA architectures. Horizontal axis correspond to average number of false positives produced on an image, vertical axis is a fraction of missed objects (the lower the better)

The pseudo-HDR mode is further enhancing overall detection accuracy, by capturing multiple images with different exposition time. The accuracy improvement depends on certain light conditions. In the case of standard light conditions, it shows improvement of approximately 5% in detection accuracy**.**

Our architecture is based on Full HD CMOS sensor. The object detection can be configured to process Full HD images on over 30 FPS (UC12.2-R001). For practical reasons we process 720p image. In this configuration, we can process framerate at over 60 FPS. This is beneficial for HDR capture mode (UC12.2-R010, UC12.2.-R011), where we capture multiple exposures and merge detection results, producing 30 FPS output. We do not explicitly merge exposures to produce HDR image, though. The HDR mode improve robustness of target applications with challenging light conditions. Our architecture

implements detector based on WaldBoost training algorithm (UC12.2-R010), which uses only a narrow stripe of image located in FPGA BRAM. All scales are calculated on-the-fly, and the detector produces results in a stream fashion (UC12.2-R008).

HIB
The evaluation of the proposed system was done for the overall metric of energy consumption as measured on an external 5000 mAh battery that powered the system.

Description of test	Description of results
Application ran a looped execution of video processing tasks. Low battery strategies were: • **lower FPS** execution (15fps and 10fps compared to the original 30fps) when battery was under 15% and • **lower resolution** (320x240 versus 640x480) when the battery was below 5%	With No adaptivity algorithm: *152 minutes* With Adaptivity algorithm: *183 minutes* (**20.4%** duration gain)

Table 6: Description of test and results

3.4.2 Assessment of impact

eVision
With the methodology used in this project and current FPGA technology, companies can build smart cameras for nice markets and non-consumer applications, without huge engineering cost or production cost. The downside is that this methodology is still highly dependent on the selected HW architecture. Nevertheless this will enable us to build functions and features into camera systems, that have been executed by external workstations in the past, and decrease cost for development for the final system.

Siemens
Filed project proposal around a network of smart cameras as a use-case as a foundation for "*freakier*" products within an Siemens internal project innovation fund called Quickstarter, very similar to www.kickstarter.com, but Siemen internal only. Funding decision is pending.

AIT
We developed a new vision processing middleware that helps to overcome problems of current surveillance systems and their QoS management, especially when hardware and software resources are limited. An important advantage of the middleware is that it can carry out reconfigurations on a graceful degradation to sustain a QoS and to prevent system failures. This characteristic is of great interest in many surveillance applications.

ITI
By using the developed algorithms it is possible to perform face detection in a wide variety of devices, not only limited to FPGAs or PCs, and adjust the parameters (memory, cpu cores, etc, …) to best fit the selected hardware.

BUT
The developed architecture can be applied in surveillance systems, quality control, and other automated systems that can benefit from visual analysis. The main advantage of the camera is that it can transmit only data interesting to the target application (license plates, faces, etc.), instead of the whole image. Such a camera works as an independent unit which requires only low bandwidth (only selected data are sent)

and very low power (can operate on batteries or solar energy). Such a camera is useful when many places need to be monitored or on remote places with bad connectivity and unavailable electrical power.

HIB

By using the proposed design process and run-time adaptive technology, we have demonstrated that we can easily extend the battery life of the proposed application (which is very related to a mobile access control system proposed by HIB and which is limited by battery life) to up to over 20% over the original. This is of course further optimisable with additional strategies for reducing the energy consumption (e.g., different encoding quality of the MPEG-4 algorithm). This result will be proposed to the product team at HIB for exploitation purposes.

4. UC12.3 – Medical imaging

In the healthcare domain, very large images need to be processed. Complex image processing is necessary to keep all image information available, while at the same time increasing the information level of the images. As very complex and configurable algorithms are in use, it is important to be able to develop the applications independent of the underlying hardware configuration, while assuring low latency for parts of the processing pipeline.

To prepare for evolution and to address the variability in products, it is necessary that the software can easily be ported to different underlying hardware configurations and the underlying heterogeneous hardware resources are transparently managed without application developers' intervention. Dynamic analysis tools will automatically detect software defects and a runtime analysis tool will enable high level partitioning decisions.

4.1 Introduction to the use case and its evaluation criteria

4.1.1 Magnetic Resonance Imaging

Figure 8 shows a typical clinical workflow for creating a magnetic resonance imaging scan for a single patient. Based on the exam request received from the referring physician, the radiologist determines in the morning which scan protocols should be used. Somewhere during the day, the real MRI exam is planned, which takes 20 to 45 minutes. Only during this time-frame the patient is at the MRI scanner. Once the scan is done the patient is dismissed to either his room or even to home in case of an out-patient. Though some image processing is done during the scan time and a first evaluation is made by the operator, the real review of the examination data by the radiologist only takes place after the patient has left the scanning room. An example of processing and post-processing is given in Figure 9.

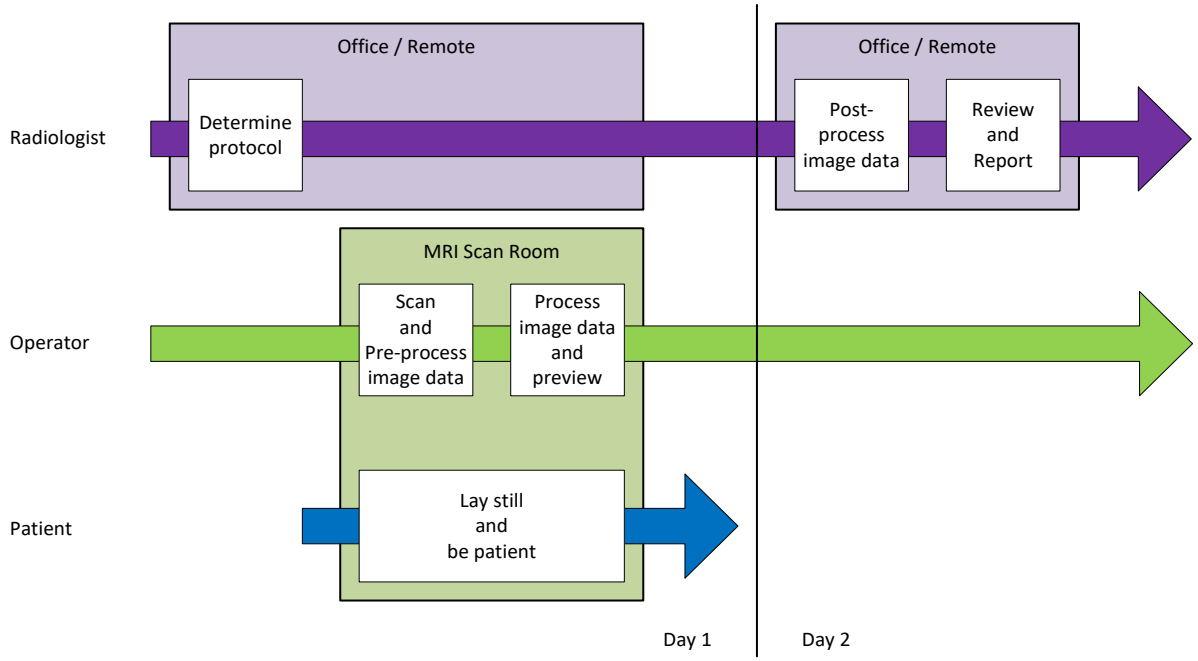


Figure 8: Clinical workflow (typical example)

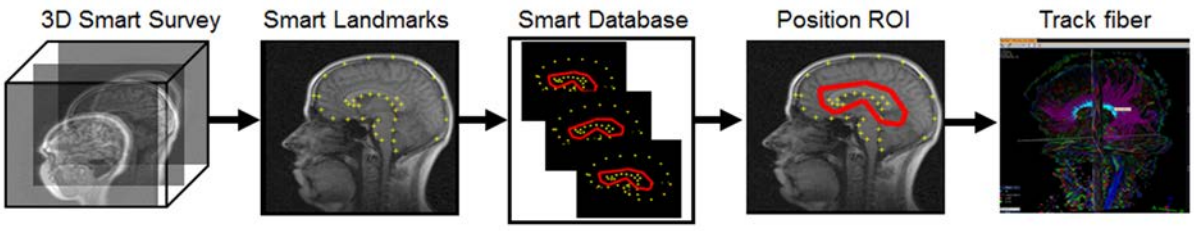


Figure 9: Processing and post-processing example (Smart DTI fiber tracking) [8]

It is only at the moment when the radiologist is analysing the data, that it becomes clear whether or not the data is of sufficient quality to be able to perform a diagnosis on it. Even though the operator is performing a first check on image quality, it still requires expert eyes of the radiologist to qualify the final image quality. Also in the case of post-processing, which is performed after the exam has been conducted, the quality check is performed well after the patient has left the scanner (and likely the hospital). Problems in data acquisition may not be visible to the operator (which does basic checks on quality), yet may prevent the radiologist to draw conclusions from the post-processed image. If the data is of insufficient quality, he has to order a rescan, which is both costly and puts an additional burden on the patient.

The solution to this problem would be to execute post-processing already at the MRI scanner. By performing more advanced checks that can be performed by the operator using post-processing, the quality check can be performed while the patient is in the scanner. If quality is proven to be insufficient, the scan can be done again. Multi-core computing is a pre-requisite for being able to perform such post-processing on the scanner console, and the mixed criticality of such as system is the main hurdle to be taken with EMC2 technology:

- Magnetic Resonance Imaging scanners create images based on spin physics. These physics dictate that latencies during scan and image pre-processing are very small (ns to µs).
- The response of the scanner should give a real-time feel to the operator at all times (ms), and image processing should not cause timing problems for the very low latency activities.
- Post-processing currently can be done based on a time/cost-value base. When this process is run on scanner hardware, the requirement becomes similar to processing.

In a state-of-the-art MRI scanner, the "host" computer runs the main process framework. In addition, it runs the user interface and various background tasks for scanning and communication control. During a scan, a real-time "DAS" computer takes over and ensures the generation of time aligned raw imaging data. Highly computationally intensive processes are required for reconstruction the raw data into useful diagnostic imaging data. Therefore, the reconstruction processes are executed on a separate "recon" system. Off-line the images are investigated by a radiologist on a "viewing station". An overview is given in Figure 10 (left).

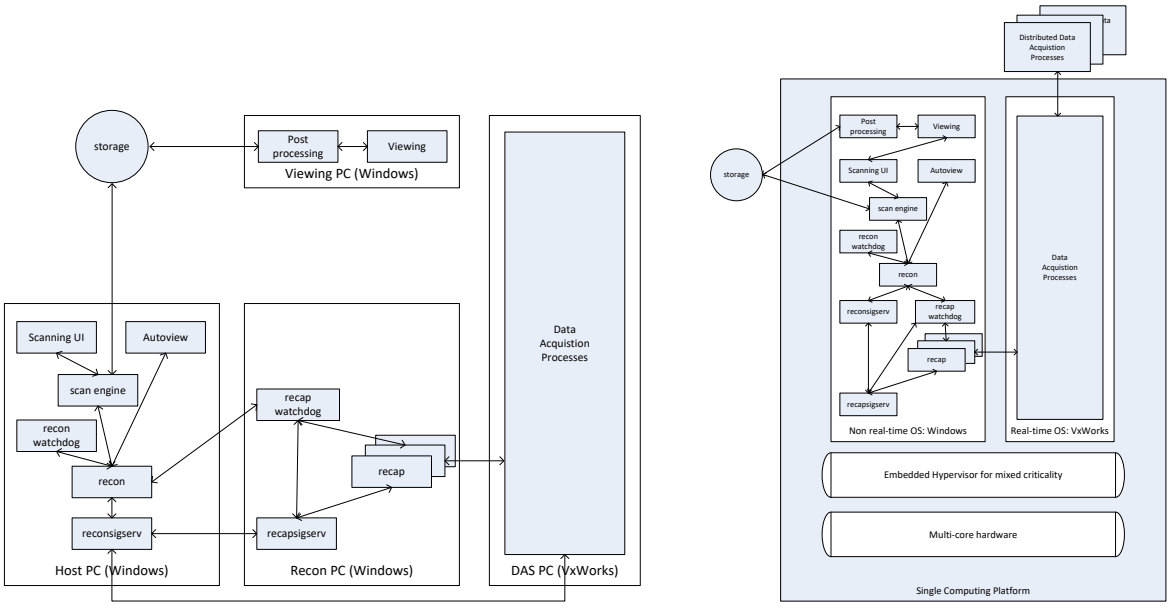


Figure 10: State-of-the art (left) and EMC² target (right)

The target of the EMC² project is to merge multiple hardware systems (host, recon, data acquisition, and post-processing/viewing) on a single hardware system Figure 4 (right).

4.1.2 Dynamic defect detection

The Medical Imaging use case evolves from a large monolithic code base run on multiple computers towards a modular approach for application on a multi-core platform. From literature, it is known that errors are part of large code bases and that new errors will be introduced when porting to a multi-core environment.

Instead of just analysing the new source code statically, the source will be compiled into a special internal representation for analysis and executed in a virtual machine of the target architecture. This means the dynamic analysis engine has full visibility into everything that is going on inside the program. Based on the dynamic program analysis, the engine will report detected software defects.

4.1.3 Runtime analysis of data communication and memory access

To address the mixed criticality dynamic analysis tooling for optimizing deployment of algorithms on heterogeneous systems will be demonstrated. This tooling will map memory access of modules and identify communication channels between modules.

4.1.4 Evaluation criteria

The evaluation criteria for the Medical Imaging use case are:

- Post-processing and viewing of post-process is possible at the MRI scanner without hampering the low-latency processes and not disturbing the real-time control by operators.
- The dynamic analysis tool will run on a Windows platform and successfully detect software defects.
- The runtime analysis tool will enable high level partitioning decisions taking communication overhead into account.

4.2 Overview of the use case implementation, its limitations, and achieved features

4.2.1 Magnetic Resonance Imaging

The first integration step combined “host”, “recon” and “DAS” as shown in Figure 11 (left) on a standard HP computer with a 6-core processor with hyperthreading (12 virtual cores) and 64 GB RAM. This prototype employs a type 1 embedded hypervisor designed for mixed criticality and a very small footprint, minimal latency, and optimizations for maximum performance. On top of the hypervisor the real-time operating system VxWorks is installed (which is identical to the existing operating system of the “scan” computer) and the non-real-time Windows 7.0. Since integration testing showed that that our requirements were not met with new state-of-the-art hypervisor technology, a second prototype was built on which the “host”, “recon” and “viewing” processes were successfully integrated.

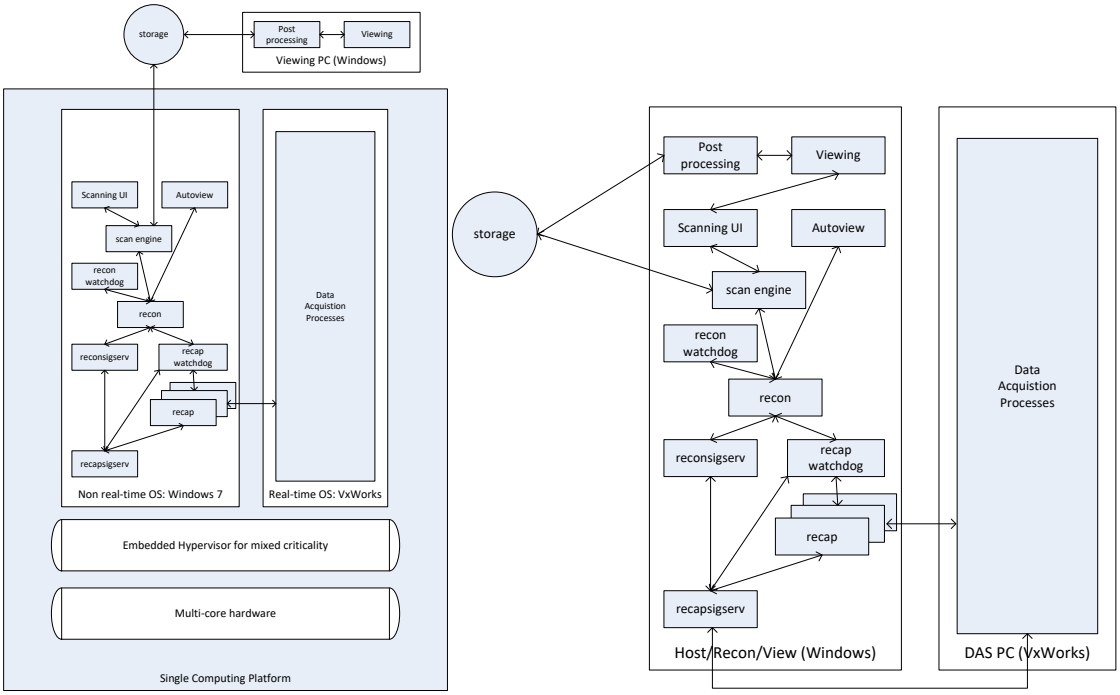


Figure 11: Final prototypes. Left: Combined Host/Recon/DAS. Right: Combined Host/Recon/View.

4.2.2 Dynamic defect detection

Vector Fabrics tool Pareon Verify for software verification has been designed to separate the execution and analysis hosts. This key feature allows for analysis of resource constraints platforms, whereas heavy computations for dynamic program analysis take place on a powerful server machine. Furthermore, the dynamic program analysis was extended to detect software defects through advanced memory tracking. For example, consider the following C program:

```
#include <stdio.h>

int a[2] = {0, 1};
int b[2] = {2, 3};

int foo(int x)
{
  return a[x];
}

int main(void)
{
  printf("a[2]=%d\n", foo(2));
  return 0;
}
```

The array access a[2] past the a array boundary is often not detected by other analysis engines, because it falls into a valid memory location of the b array. However, this mistake leads to program malfunctioning.

4.2.3 Runtime analysis of data communication and memory access

TU Delft developed the MCProf tool, which is a runtime memory and communication profiler. It generates detailed application profiles, in terms of memory access patterns and data-communication at function and loop-level granularity. It is based on Intel Pin Dynamic Binary Instrumentation (DBI) framework. MCProf performs instruction-level instrumentation to track memory reads and writes by each instruction. Furthermore, routine-level instrumentation is utilized to keep track of the currently executing function. These are tracked by maintaining a call-stack of the functions executing in the application. To track the dynamic allocations in the application, image-level instrumentation is utilized to selectively instrument library images for memory (re)allocation / free routines.

The tracked memory reads/writes are associated with parts of the source code, depending upon the selected granularity, i.e. functions, loops or other marked regions in the source code. In this way, a producer-consumer relationship is established between functions/loops/objects in the source code and reported in the form of a communication graph. This production-consumption relation is expressed by edges in the graph, where the colour of the graph shows the intensity of the communication. Furthermore, the generated graph also shows percentage of the dynamically executed instructions and the number of calls to each function. In this way, communication as well as the computation intensive parts of the application are shown in the generated graph.

To automatically utilize the output generated by MCPRof, a complete toolchain was developed to extract the available parallelism in an application from its sequential implementation. Figure 12 shows the block diagram of this framework.

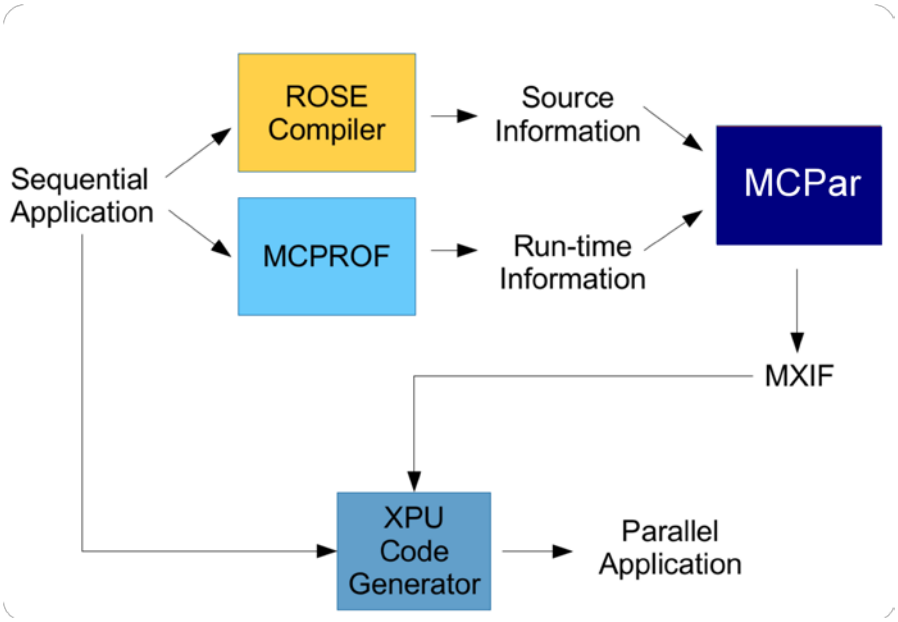


Figure 12: Block diagram of MCPar which is an MCPRof based application parallelisation framework

The toolchain can extract the following types of parallelism in the application:

- Coarse-grained: at function call level
- Coarse-grained: at the loop nest level
- Fine-grained: at loop iterations level

To extract this information, source-level information generated by Rose compiler is combined with runtime information generated by MCProf. This is shown in Figure 8. For this we extended MCProf to generated extra information, for instance the call graph of the application at various granularity levels.

MCPar combines the source and runtime information to parallelize the application at various granularity levels to generate parallel representation of the application.

Parallelism in Gaussian_smooth function	Parallelism in derivative_x_y function

`sequential gaussian_smooth_183_7`
`parallel_for gaussian_smooth_73_11_pf`
`parallel_for gaussian_smooth_71_12_pf`
`parallel_for gaussian_smooth_79_14_pf`
`parallel_for gaussian_smooth_77_15_pf`
	`parallel derivative_x_y_186_19`
	`parallel_for derivative_x_y_63_20_pf`
	`parallel_for derivative_x_y_61_21_pf`
	`parallel_for derivative_x_y_67_22_pf`
	`parallel_for derivative_x_y_65_23_pf`

Table 7: Simplified output generated by TU Delft tool chain showing extracted parallelism inside Canny Edge Detection application.

4.3 The intended innovation

Application level

- Viewing of post-processing result on the MR Console (while the patient is still at the scanner);

Technology level

- Single hardware platform for execution of all MRI (mixed-criticality) processes;
- Tool for dynamic code analysis on a Windows platform to successfully detect software defects;
- Tool for runtime code analysis supporting high level partitioning decision taking in to account communication overhead.

4.4 Evaluation results

4.4.1 Evaluation of outcomes from the use case

Overall test results show that the functionality requirements are met on the combined host/recon/viewing prototype. Performance criteria are met, except for applications with very high-demanding reconstruction needs. An increase in cores is expected to solve this issue, but currently these are not available in standard hardware.

Also, the combined host/recon/scan processes have been integrated on a single platform. Scanning is possible without scan aborts. However, 1 out of every 1000 samples exceeds the maximum allowed peak latency. Within the EMC² project the root cause could not be identified yet. It is assumed that there is a mismatch between the architecture of the hypervisor and the Intel VT platform, when overclocking of the processor occurs during run-time, because the MRI application requires a very stable clock (i.e. needs a different trade-off between robustness and speed). In addition, runtime limitations in the VxWorks real-time operating system have been identified which hamper parallel execution of recon and scan processes (which shared memory access for high speed data transfer). RTLinux could be a better choice, which has the additional advantage of being more compatible with the analysis tools developed in EMC².

Vector Fabrics analysed various medical imaging software with their tool Pareon Verify. Both data races and out-of-bounds have been detected by Pareon Verify. Bug reports were submitted and fixed by code owners. Here is a selected list of errors discovered with corresponding bug report links:

1. Visualization ToolKit: a data race, http://www.vtk.org/Bug/view.php?id=15903
2. Visualization ToolKit: an out-of-bounds access, http://www.vtk.org/Bug/view.php?id=14997
3. DCMTK (DAICOM): null-pointer dereference, http://git.dcmtk.org/?p=dcmtk.git;a=commit;h=24e9c0b25cf994f62b0fe5ecbcd888dd7c526816
4. GDCM (DAICOM): 4 various out-of-bounds and other errors detected, http://sourceforge.net/p/gdcm/mailman/message/34726458/
5. Uninitialized memory reads in the OpenCV unit test UMat.async_unmap, https://github.com/Itseez/opencv/issues/6102
6. Memory leak detected in the OpenCV unit test Flann_LshTable.badarg, https://github.com/Itseez/opencv/issues/6131

Performance slow-down due to analysis was initially quite large and around 1000x. By applying a series of various performance optimizations in Pareon Verify tools, it was reduced to 100x.

MCProf generates the execution profile of an application and presents it as a flat profile. Apart from the execution profile, MCProf also lists the memory intensive functions and objects in the application. This highlights the compute intensive functions in the application and the objects responsible for most of the communication in an application. Apart from the flat profile, MCProf also generates the data-communication graph of the application, showing the flow of data in the application. Figure 13 shows an example graph for Canny image processing application. Functions in the application are represented by ovals, with the percentage of dynamically executed instructions and number of calls to this function. The objects allocated in the application are represented by the squares with their sizes. The edges represent the communication, where intensity of communication is represented by color of the edge. Another interesting feature of MCProf is that it can also show the information about loops or any other marked regions in the application.

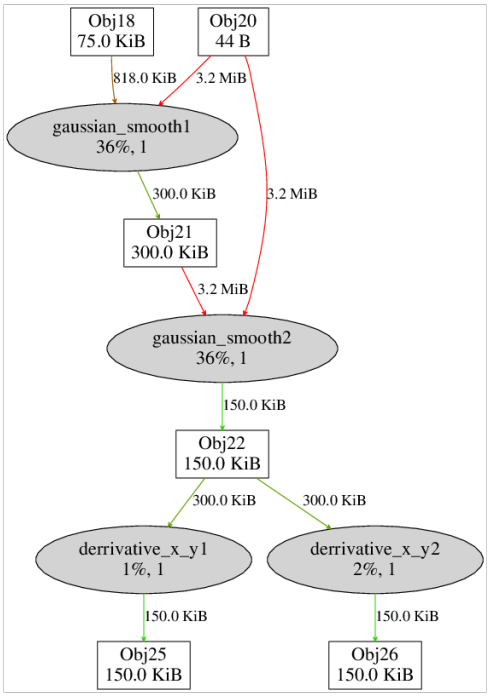


Figure 13: Simplified data-communication graph generated by MCProf. Ovals represent the function in the application and Squares represent the allocated objects.

To give the reader an idea about the overhead of the MCProf tool and its comparison with state of the art, we present both the execution-time and memory-usage overheads for a range of applications. These results are depicted in Figure 14 . Results with MCPROF legend are overheads, while providing the common basic information which Pincomm and Quad can also generate. Whereas, MCPROF report overheads of complex engine while generating the detailed data-communication information with stack recording and object detection. Mean overhead results are also depicted in these figures. These results show that MCProf has, on average, an order of magnitude less execution-time and memory-usage overheads.

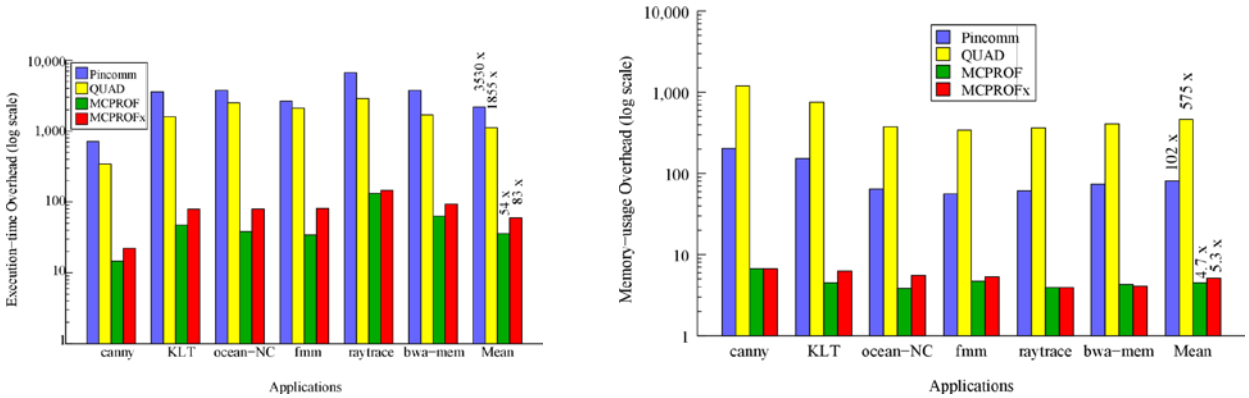


Figure 14: Execution-time and memory-usage overhead comparison of MCProf with state of the art tools for a variety of benchmarks

4.4.2 Assessment of impact

For low- and mid-end systems Philips has productized the first EMC² prototype, which integrated Host and Recon on a single hardware platform. When hardware with more cores becomes available, this will be extended to high end systems. Feasibiltiy of viewing on the console has been proven.

Though Vector Fabrics reported great results with Pareon Verify they had to file for bankruptcy during the last year of EMC². TU Delft will continue the development of MCPar toolchain.

5. UC12.4 – Control applications for critical infrastructure

5.1 Introduction to the use case and its evaluation criteria

5.1.1 Description of the use case

ABB developed as a use case an application covering the business domain of power system control. The main target was a "Cooling System for Transmission Plant" (CS_UC) - Figure 15. The application is a closed loop control system, where a number of sensor elements and actuators are connected by various interfaces. The system performs relevant actions depending on the input signals, the internal system state, the configurable logic, and possibly on operator commands. The system is required to perform a variety of computation intensive operations, with very high real-time requirements, on data coming in concurrent streams. More information on the initial specifications to be found in deliverable D12.1.

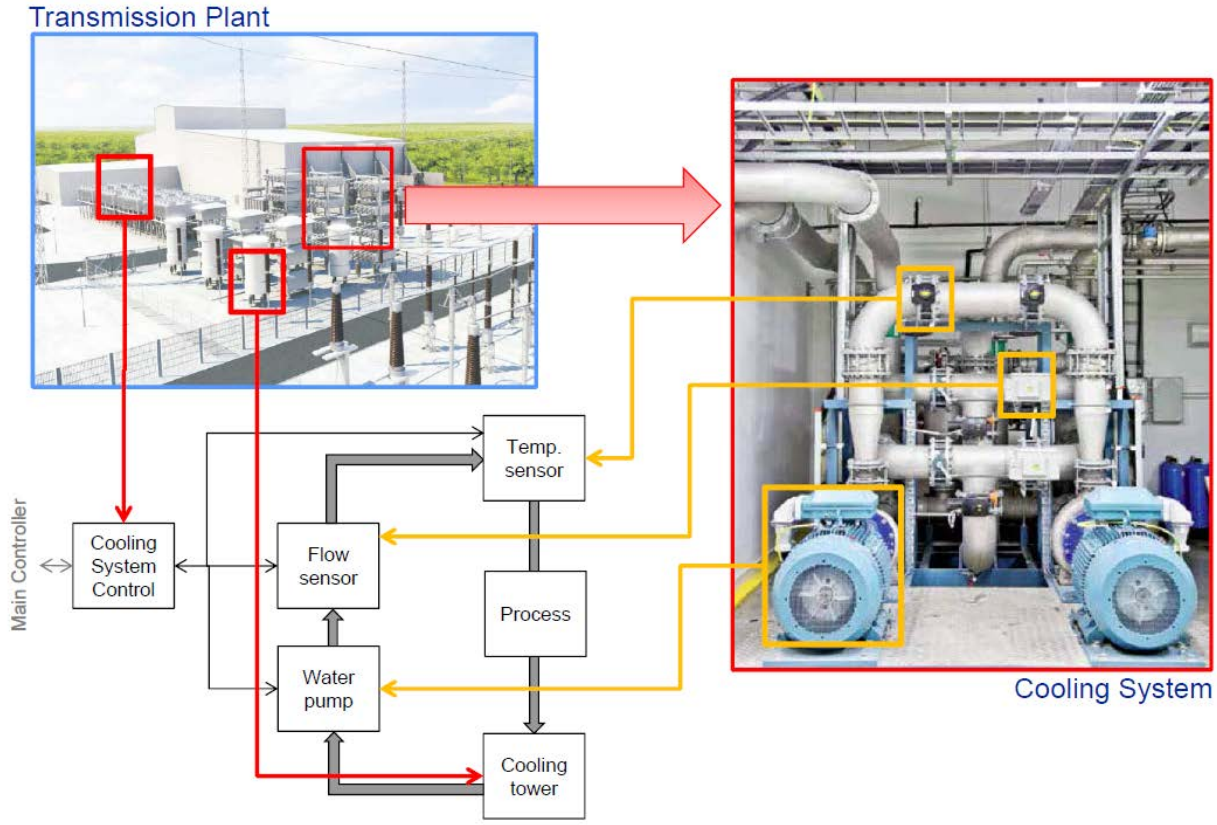


Figure 15: Context for the Cooling System for Transmission Plant Application (CS_UC).

The use case objectives focus on the creation and execution of a tool-chain to support application development, together with multi-core design aspects such as application partitioning and mapping. Thus, in brief, we target to (semi-) automatically provide an (almost) optimal solution for multi-core SW/HW partitioning and mapping, and to (semi-) automatically exchange tools at different stages in the design flow within the used tool-chain.

5.1.2 Solution - Tool-chain realization

The envisaged tool-chain is to be built on the basis established within the iFEST project, enabling tool replacement and tool independence. In addition, proper tools for specific tasks are to be employed (such as Microsoft TFS / Visure Requirements for the requirements stages).

Several artifacts are to be developed within the project.
Figure **16** presents a generic image of the iFEST tool framework, where tools are dynamically interconnected via tool adaptors - TA. The focus in the current project is to develop several such TAs for

existing tools, but also to build additional tools, based on the project development (especially WP5 activities).
At this moment, several tools have been identified as suitable for use in the application development:

- Microsoft Team Foundation Server – to be used for requirement storage, code storage, collaborative artifacts. Tool adaptors for various functionalities to be built during EMC2.
- Visure Requirements – to be used for development and storage of requirements. Tool adaptor to be updated during EMC2[2].
- HiDraw – a graphical design tool, used to build the application at various levels of abstraction (code to system levels). Tool adaptor to be built during EMC2.

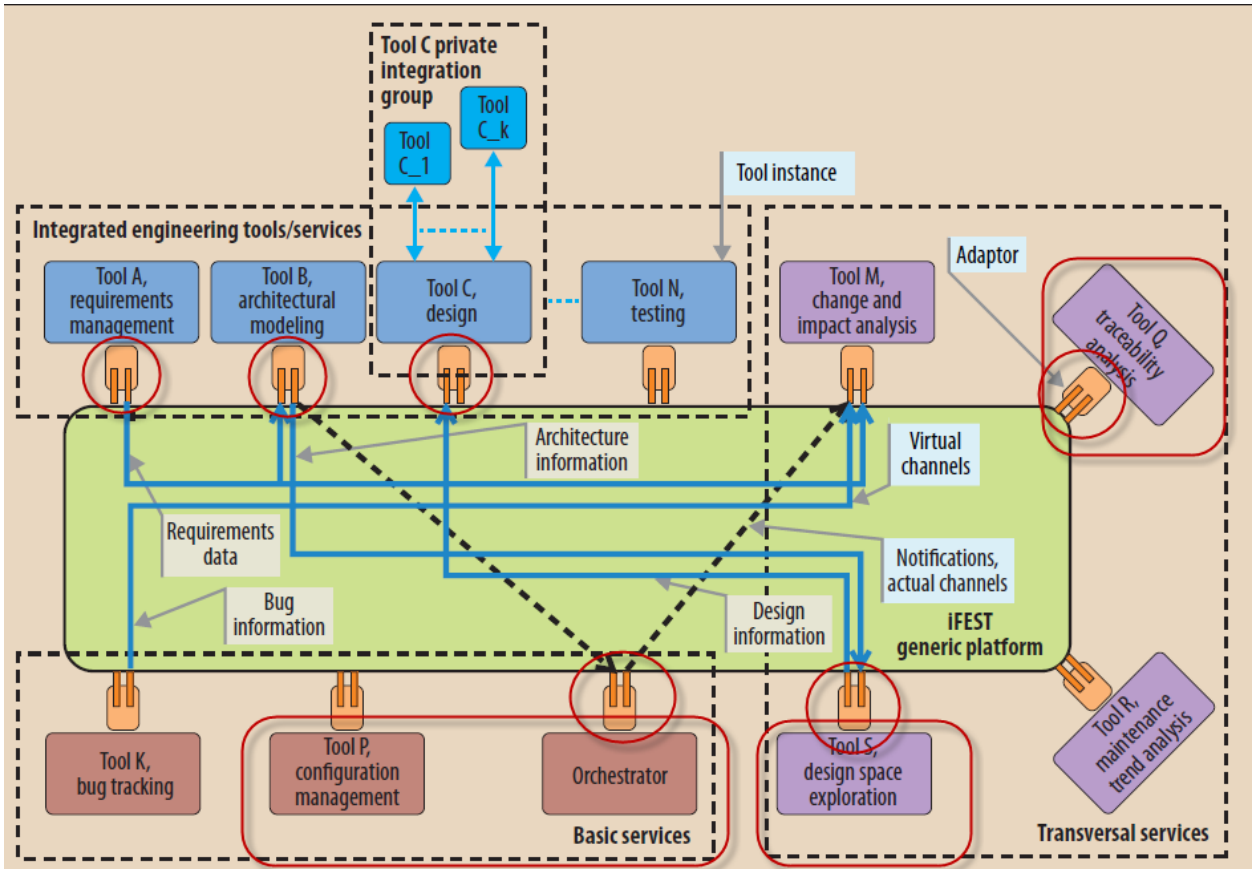


Figure 16: The original iFEST tool platform, showing the development focus points within EMC2.

The tools' respective TAs were developed during the project. An additional *Orchestrator* tool provides ways for transparent (source / destination independent) tool interaction.

The specific goals we want to achieve here can be identified along the following lines:

- Integrate tools to achieve higher engineering efficiency
  - Reduced design time
  - Increased quality
  - Reduced time to look up information
  - Reduced tool switch time
  - Traceability of artefacts
- Extend the life-cycle of system engineering tool chains

[2] Visure eventually had to leave the project.

- Tool independence
- Easy replacement of tools
- Avoid vendor "lock-in"

5.1.3 Evaluation criteria

A set of metrics was followed up after the realisation of the supporting infrastructure and its installation in the actual context, addressing:

- IFT (Information Find Time): The time needed to find information of other tool's artefacts (such as requirements related to a given design module). Measured in seconds (for actual values) or given as relative improvements (%).
- TST (Tool Switch Time): The time spent on switching between tools. Measured in seconds (for actual values) or given as relative improvements (%).
- TN (Trace Numbers): The number of traces (links between different kinds of artefacts, e.g. design elements linked to requirements). Measured in number of items, or coverage (%).
- TI (Tool Independence): Level of tool independence. Measured by the required knowledge of a tool operator about the other tool(s) in the chain. Values as percentage (100% = full independence, 0% = full tool interdependency).

5.2 Overview of the use case implementation, its limitations, and achieved features

5.2.1 Description of the final prototype

The final prototype was deployed within the actual development network, and it contained all the artefacts realized in the project. One user was possible to employ in actual context for the assessment of the results.

An orchestrator (including name translation and notification services) are deployed on a virtual machine accessible through local set-up, while respective tool adaptors (for the design tool HiDraw and the Microsoft TFS modules for requirements and version control) were located on the developer's machine and on the "TFS Server", respectively. An overview of the architectural perspective is shown in Figure 17.

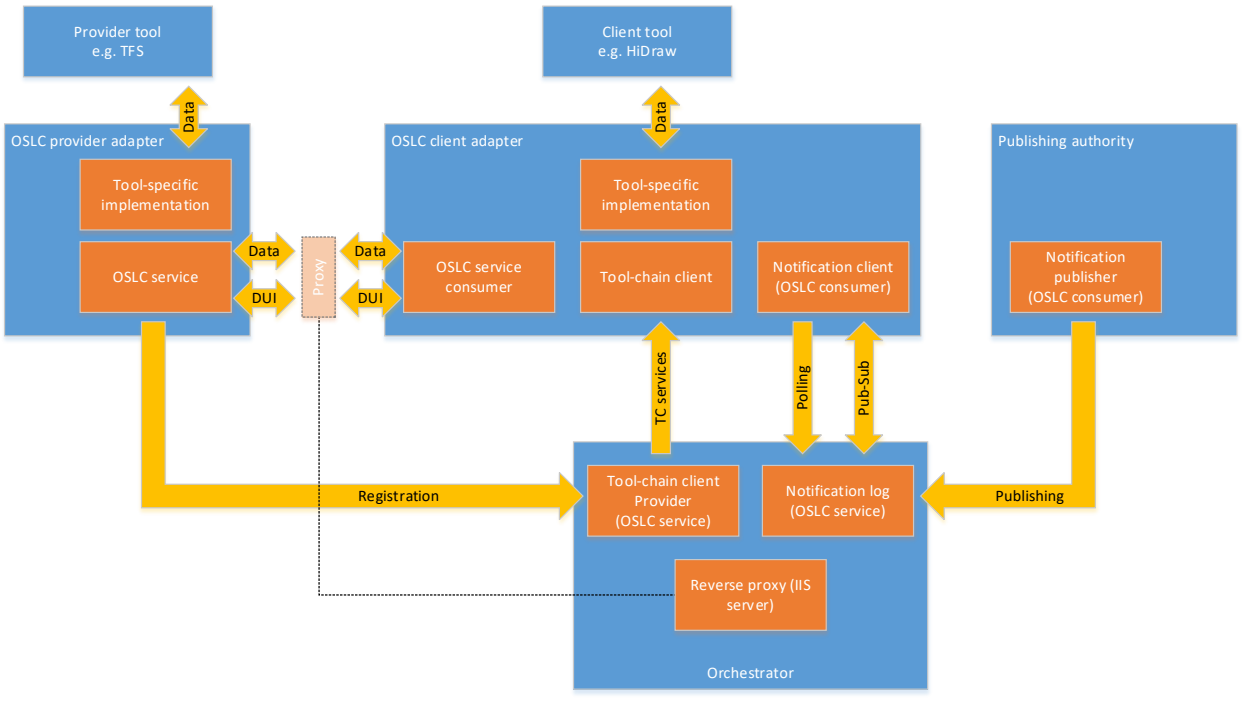


Figure 17: Overall architecture of the implemented system.

Due to the Visure exit from the project, an important aspect of the approach – assessing tool replacement scenarios – was not possible to implement.

An additional set-up was used in an experimental environment, where multiple virtual machines were used to allow the usage of 2 HiDraw users (numbers limited by computer features).

5.3 The intended innovation

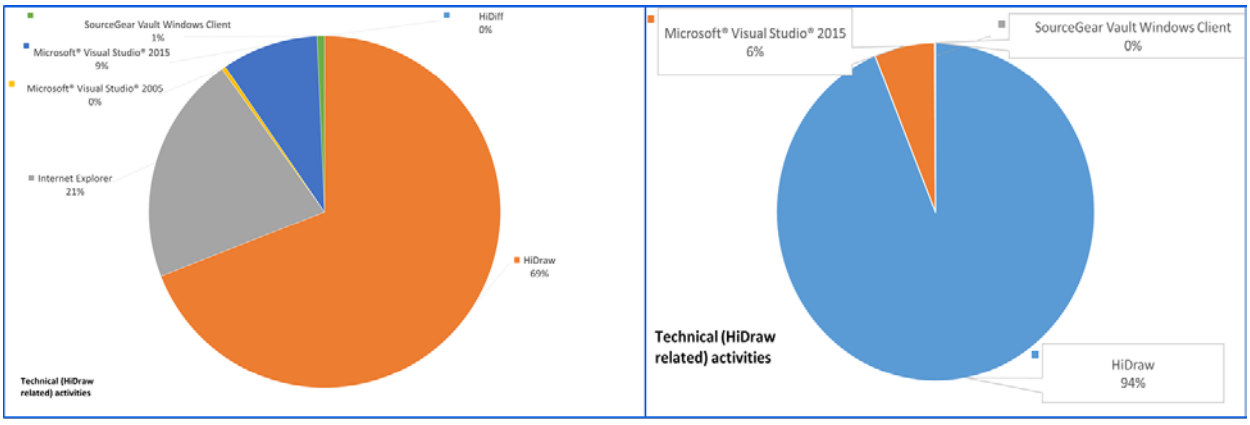
The innovation in this case is based on the transparency of tools used at different stages of the design activities. This is partly demonstrated by the eventual elimination of one of the tools: Internet Explorer is present in the "state of practice", but absent in our latest implementation. The separation realized with the help of the Orchestrator also supports this approach: The Orchestrator implements a HTTP reverse proxy that hides the real URL of a tool, and it rewrites all OSLC resources to replace tool-specific URLs with proxy ones. Clients access the proxy, which, transparently, redirects them to the targeted tool.
Another part of the demonstrator was intended to illustrate the exchange of requirements tools. However, it was not possible to execute this, since Visure Solutions left the project.

5.4 Evaluation results

5.4.1 Evaluation of outcomes from the use case

Under the measurement exercises, final values were identified for the first three of the above metrics (section 5.1.3).

The final results show an important decrease of actual work time within technical tools, reached due to the integration capabilities. TST and TN final values have defaulted to (approx.) 0s, respectively 100%. A graphical representation illustrating cumulated values for IFT and TST and the impact on technical work duration is given in Figure 18.



a) Technical developments without an integrated tool chain

b) Technical developments with the integrated tool chain

Figure 18: Results before a) / after b) of the taken approach.

5.4.2 Assessment of impact

The above figure tells us the usage of the tool chain increases by a quarter the possible amount of technical work, reduces (by a third) the technical supporting activities (in this case requirements analysis) and eliminates additional tools used today to navigate between tools - as Internet Explorer, with usage data as in Figure 18 a).

The results support further either a more efficient and focused development context, or that more efforts can be deployed in the same amount of time within the project.

We eventually can conclude that all the goals specified in section 5.1.2 have been reached, and, except tool replacement, also demonstrated.

6. UC12.5 – Railway applications

6.1 Introduction to the use case and its evaluation criteria

The Thales Austria (TAT) use case is concerned with virtualisation of the TAS Platform, the implications for the safety health monitor and means to exploit the gained resources in safety critical railway applications.

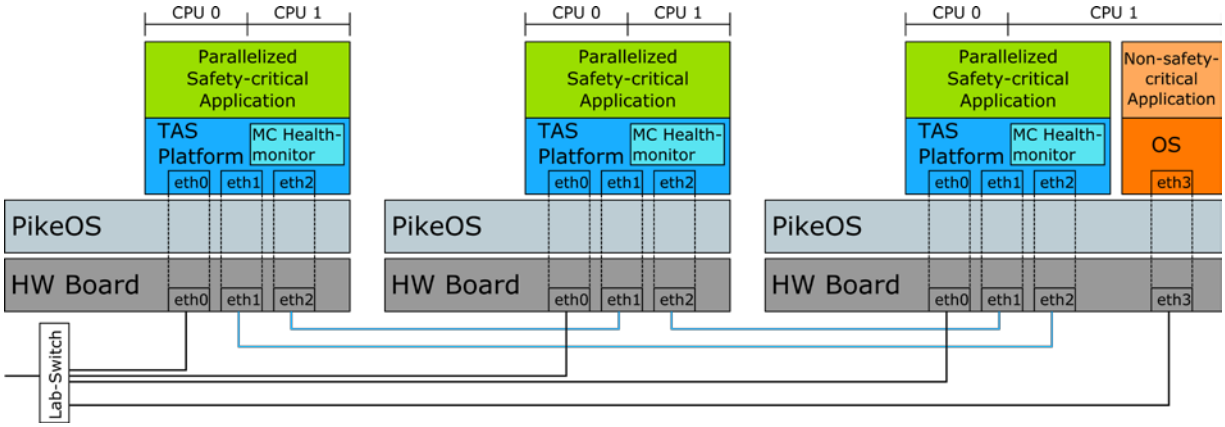


Figure 19: TAS Platform on multi-core demonstrator architecture from [D12.3]

The TAT prototype features the abovementioned features, in the sections below we will give insight on the evaluation criteria and implementation status of each work package, as well as references to project requirements.

Evaluation Criteria for Novel Programming models: (requirements UC12.5-R0100 and UC12.5-R0110)

Computationally intensive tasks in railway-control can benefit from additional performance by using parallelization. However, integrating parallel applications in synchronized replicated systems requires the applications to be replica-deterministic. A programming environment supporting replica determinism and parallelism is a way to reach those goals.

We have identified the following desirable properties of a deterministic framework:

- Performance of multithreaded benchmark implementations shall provide an improvement over serial (single threaded) versions of the programs.
- Fully transparent frameworks are preferred over frameworks which require the programmers' intervention.
- Higher level of abstraction of parallelism in programming interfaces is preferred over low level, architecture dependant construct (e.g., CilkPlus is preferred over direct pthreads programming).
- Timing behaviour shall be analysable.

Evaluation Criteria for health monitor: (requirement UC12.5-R0120)

The health monitor shall be evaluated according to the following criteria:

- Health monitor runs on both cores (i.e. checks all resources)
- Sync for testing shared resources (L3 cache, memory)

Evaluation Criteria for virtualisation: (requirement UC12.5-R0130)

TAS Platform virtualisation on top of PikeOS shall be evaluated according to the following criteria:

- Run TAS Platform next to a partition with non-safety-critical application

Ensure that the PikeOS scheduler does not interfere with the TAS Platform synchronization mechanism

6.2 Overview of the use case implementation, its limitations, and achieved features

Novel Programming models:

Deterministic MultiThreading (DMT) approaches in combination with high level multithreading frameworks may provide the expected guarantees with respect to replica determinism and desired performance. Integration of DMT into the TAS Platform seems to be feasible in the light of the published results in this area. We classify the compatibility of DMT related work to the TAS Platform in three compatibility classes.

Compatibility Class A refers to implementations that require little of no source code modification, these systems can be compiled into most multithreaded applications with little or no modification to the original code.

Compatibility Class B refers to works whose application programming interface provides a class A compatibility, but require minor additional software to be integrated into the environment such as kernel modules, software daemons, customised compilers etc.

Compatibility Class C refers to implementations which are definitely invasive to the TAS Platform software environment. For example, a system that requires a customized hardware, operating system or that is not compatible with POSIX threads (that includes non-C programming languages).

The following table presents a classification matrix of the 10 most relevant published works which have been examined during this research:

System/Paper	Class
DMP	C
CoreDet	B
Kendo	A
Grace	A
Determinator	C
dOS	C
DThreads	A/B
RFDet	A
Consequence	B
Parrot	A

Table 8: Tool/Method classification for deterministic multithreading

Most of the analysed works in class A or B rely on a “drop-in” replacement of the pthreads library. This makes it possible to leave a parallel application without modifications, and just link or re-compile together with the deterministic library. This is possible by letting the DMT implementation to be fully compliant with the POSIX threads library. Further investigation needs to take place to better understand the real-time properties of these systems.

Health monitor:

We implemented a health monitoring application that periodically tests all memory cells of the equipped RAM. The application performs a destructive memory test (write followed by read-back) for all reachable addresses. Synchronization between all cores is needed to avoid concurrent access of shared resources during the test process (i.e. RAM memory and shared caches). Our first approach implements a synchronization algorithm that executes a destructive memory test on one core, while waiting for the successful test result on all other cores. This implementation ensures that no core but one is concurrently testing a memory cell.

Virtualisation on PikeOS:
Separation of partitions on top of a shared hardware infrastructure enables the integration of several legacy applications. Apart from the general separation concern, i.e. non-interference between partitions, this separation mechanism must not affect or even prevent the successful execution of the TAS Platform synchronization mechanism. To ensure this property the TAS Platform directs the PikeOS scheduler. Consequently, one of the already identified limitations of this implementation is the synchronization precision compared to a system without hypervisor and time partition. Both these mechanisms increase communication delays and the jitter in scheduling of TAS Platform, which in turn only allows more relaxed timing requirements by the safety-critical applications.

Achieved features

- Classification for applicability of deterministic multithreading methods
- Solved synchronization on health monitor memory test for arbitrary number of cores
- Solved concurrent test execution
- Successful virtualisation of TAS Plf on PikeOS

6.3 The intended innovation

In the railway domain, there is a large use of COTS HW due to usually low quantities of sold products as compared to e.g. automotive. Still there is a customer demand for 20+ years of application support, which has to be leveraged through the TAS Platform architecture as depicted in Figure 20.

Soon, multi-core processors will be the only COTS processors available. Still, the switch to multi-core must not affect the safety critical application, which requires a new safety concept for the hardware and the OS. This comprises the main challenge for the railway domain and one innovation in this project.

Another challenge is the exploitation of gained resources, i.e. making future railway applications fit for the hardware. This was addressed through the introduction of deterministic parallel programming models.

Finally, virtualisation tackles both challenges, as it offers a safe way to integrate multiple applications on a single HW board, but it also offers the possibility to further exploit the resources by using multiple virtualised instances.

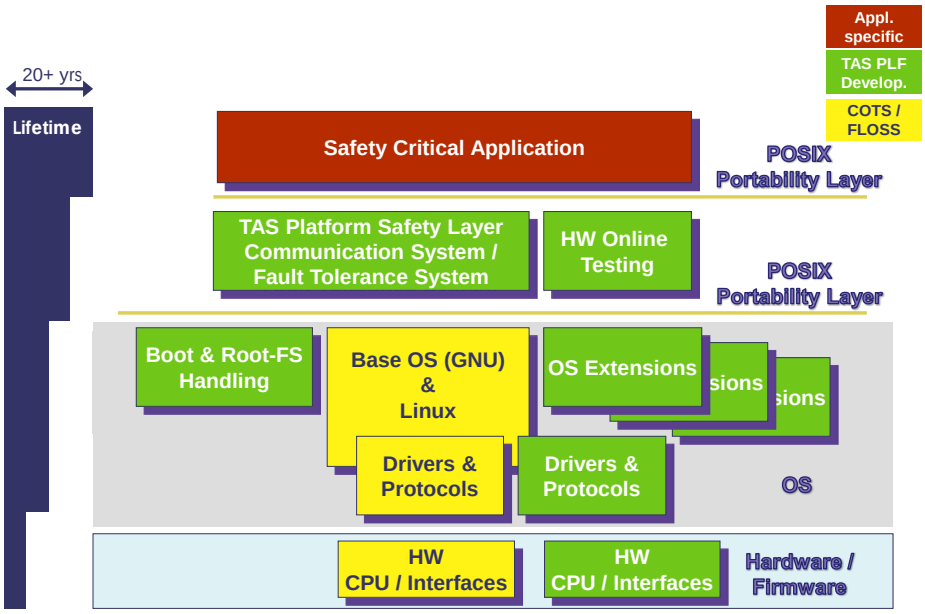


Figure 20: TAS Platform architecture and operational lifetime

6.4 Evaluation results

Below we will give a technical evaluation of the use case outcomes as well as an assessment of the impact.

6.4.1 Evaluation of outcomes from the use case

Novel Programming Models:
As a first improvement, the concept of Sparsely Deterministic Memory (SDM) has been introduced. In SDM, the programmer can choose the memory regions (e.g., variables, memory objects) that require to be treated deterministically. Data races occur, when more than one thread concurrently access a memory register, and at least one of these is a modifying access (e.g., write). If a data race occurs to a variable which is declared deterministic, the framework will ensure that the result is deterministic by enforcing a deterministic per-variable ordering of events. Combined with a deterministic mutual exclusion algorithm (e.g. Kendo [9]), SDM yields a deterministic execution scheduling which is sufficient to build replicated applications.

Second, the concept of Asynchronous Programming using Futures [10] has also been explored. A prototype of this programming model was developed in C including various scheduling techniques like work-stealing, thread pools and plain Operating System style threads. This style of parallel programming can be used to create parallel applications by explicitly specifying the flow of data of the program. In addition to that, it is also easy to migrate a sequential application to multi-core using this programming model.

Safety health monitor
The RAM/Cache test of the health monitor was improved from the last prototype, as it now supports running in parallel on all cores. To that end, a synchronization mechanism was implemented for the cores when entering the critical testing section, since otherwise the test may affect the other core's memory region as this test must run in kernel space.

Also, the test must not last longer than a few µs for each cycle (very short interrupts of the safety critical railway application running on the processor) so the synchronization mechanism had to be quick. The test's functionality is depicted in Figure 21. First, all running applications are blocked and all interrupts are disabled. Then the test is run on both cores (in kernel space) where each core selects a different part in the memory space which is tested and then restored for the applications to continue their operation.

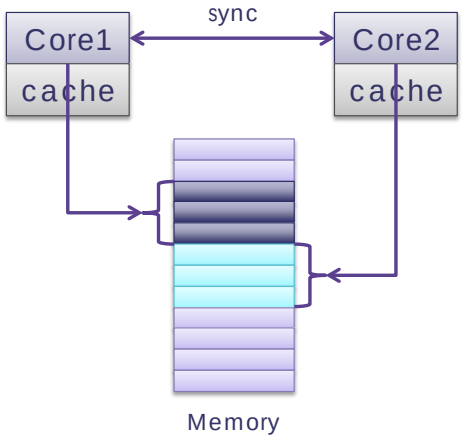


Figure 21: Multi-core synchronization for Safety Health Monitor

By using two cores instead of one we noticed a speed gain of a factor of 1.5. The reason this was not doubled lies in the overhead of the synchronization mechanism.

TAS Platform virtualisation on top of PikeOS
The core feature of PikeOS is to provide strict partitioning on top of multi-cores. For sharing of the CPUs, PikeOS has a time-sliced scheduler for encapsulation. This scheduling strategy is "as-is" unsuitable for the use of the TAS Platform 2oo3 system that implements the synchronisation mechanism in this setup within a partition of PikeOS, since it would result in unacceptably long reaction times for the safety-critical applications. With the TAS Platform on top of PikeOS prototype, it has been shown that this restriction can be overcome.

Based on the criteria for virtualisation (requirement UC12.5-R0130), TAS Platform virtualisation on top of PikeOS is evaluated according to the following criteria:

- Run TAS Platform next to a partition with non-safety-critical application
- Ensure that the PikeOS scheduler does not interfere with the TAS Platform synchronization mechanism

Concerning the non-safety-critical application, a second partition has been defined in the prototype next to that of TAS Platform and assigned a time frame for scheduling. Every time the non-safety-critical partition is scheduled, the CPU cores are not available to the TAS Platform partition. In the prototype, the non-safety critical partition is assigned 80% of the CPU time and TAS Platform 20%.

To ensure that the PikeOS time-sliced scheduler does not interfere with the TAS Platform synchronization mechanism, TAS Platform must be able to adjust the PikeOS scheduler according to its needs, while still permitting PikeOS to ensure the partitioning. To achieve this, two different scheduling schemes are defined, one while booting and the other during operation. This is illustrated in Figure 22.

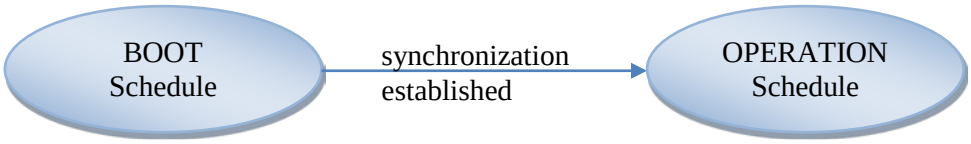


Figure 22: Scheduling states in prototype

The prototype showed that TAS Platform is able to control that the partition it is running in can be scheduled the next time the synchronization mechanism needs it and therefore systems needing the time-slice scheduling capabilities of PikeOS could also use it with TAS Platform.

6.4.2 Cybersecurity and safety considerations in railway applications

AIT looked into the issue of cybersecurity in context of the DIN VDE approach and Use Case 12.5., based on knowledge from work with the existing railway standards and evolving groups in IEC TC65 (e.g. WG20, Framework to bridge requirements for safety and security) and the joint SAE/ISO TC 22 SC 32 group for "Road vehicles – Cybersecurity Engineering". Since this analysis was started late in the project, it is not directly part of the TAT railway demonstrator. Still TAT will use the insights from the analysis for ensuring security in future versions of their TAS Platform.

Today, the relevant railway safety standards do not address cybersecurity, which is due to the fact that until recently railway information systems were rather separated from others, including communication lines. In the railway domain, safety engineering is guided primarily by a set of four standards:

- EN 50126 Railway Applications – the specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS):
  - Part 1 – Basic Requirements and generic process (CENELEC 1999, Corr. 2010)
  - Part 2: - Guide to the application of EN 50126-1 for safety (informative) (2007)
- EN 50128 Railway Applications – Communication, signalling and processing systems – Software for railway control and protection systems (IEC June 2011)
- EN 50129 Railway Applications - Communication, signalling and processing systems – Safety related electronic systems for signalling (IEC 2003, corrigenda 2010).

- EN 50159 Railway Applications - Communication, signalling and processing systems. Safety-related communication in transmission systems (2010)

A short discussion of the standards reveals that cybersecurity issues are not addressed at all in the first three safety standards. EN 50159 does not cover general IT security issues, and it does not cover IT security issues concerning confidentiality of safety-related information and prevention of overloading of the transmission system. But it addresses the issue somehow in case of open transmission cases, admitting that against different security threats an overall program has to be defined, encompassing management, technical and operational aspects. Only intentional attacks by means of messages to safety-related applications are considered.

Therefore, work towards addressing the security aware safety case is evolving. The draft standard update of EN50129 from 2016 for the first time mentions IT security explicitly as a concern, leaving no doubt that IT-security can affect not only the service, but also the functional safety of a signalling system. However, this standard does not specify the requirements for development, implementation, maintenance and/or operation of security policies or security services, for which appropriate IT-Security standards are applicable.

In the course of the standard update, work on defining harmonised IT security requirements for railway automation was started, with the goal to build on the well-known safety certification processes of EN 50129, EN 50159, and integrate security requirements based on IEC 62443. A proposal was submitted by DKE (Germany), integrating requirements from IEC 62443 in the railway standards, addressing EN 50129 and EN 50159 issues, as DIN VDE V 0831-104 “Electric signalling systems for railways – Part 104: IT Security Guideline based on IEC 62443”. This integrative approach which builds on established safety processes seems promising.

The work was brought forward to CENELEC TC9X, Electrical and electronic applications for railways, SC9XA (Signalling), by a subgroup SGA16 which was asked ”to prepare a report for SC9XA, setting out the proposed scope of a future standard on IT security and recommending whether it should be a EN/TS/TR.” The SGA16 report recommended (Sept. 30, 2016):

- to create an EN (Standard, not a Technical Specification or Report) related to IT Security for Signalling
- a scope for a NWI proposal (New Work Item)
- to embed this EN into an overall framework at TC9X

The report states, that there is no question, based on past experiences on severe cybersecurity attacks in many domains and systems, that we NEED information security, and that it is not just a technology issue, but processes and people aspect should be taken into account on an overall (holistic) level for railways.

6.4.3 Assessment of impact

The outcomes of this project impact the TAS Platform in the following ways:
Introduction of novel programming models for TAS Platform railway applications enables the programmer to use multi-core features deterministically on a higher abstraction level. Asynchronous Programming using Futures offers a particularly easy interface to write parallel programs, by specifying data-flow and computation dependencies explicitly throughout the code.

The TAS Platform enables railway applications to run in a safe environment, enabling a significant reduction of system design for those applications. The methods developed in EMC² (safety concept, health monitor) are an enabler for using the TAS Platform on future HW generations.

The TAS Platform eases revalidation and recertification of railway applications. Virtualisation solutions, such as PikeOS, further enable the integration of several (mixed critical) applications while guaranteeing their independence.

7. Conclusions

The evaluations show that all the WP12 tasks have achieved significant innovation results. Time, efforts and resources needed to develop and execute the multi-core applications of the use cases have been significantly reduced.

7.1 UC12.1: Seismic surveying by ship

In the EMC2 project's Document of Work (DoW) we stated our vision for the results as follows:
Software engineering for multi-cores is a major bottleneck in embedded systems engineering, and T12.1 will make it possible to produce software in less calendar time and at significantly reduced costs. For the development work and embedded systems targeted in T12.1, the aim is to reduce engineering calendar time to 1/10th of what it takes today, and to reduce execution time to 1/5th of that of today's systems.

The multi-core C++ code automatically generated by *m2cpp* runs 2-60 times faster than the serial MATLAB code for the prototype examples. For one of the prototype examples, a couple of hours hand optimisation shows that the generated and hand optimised code utilising multi-cores runs up to 250 times as fast as the MATLAB code running on a single core.

We estimate that developing with *m2cpp* is 10 to 100 times faster, compared to the old way of working. I.e. the engineering calendar time using *m2cpp* is from 1/10th to 1/100 of what it takes using the old way of working.

m2cpp will be very useful in the future work for the global HPC-group in Schlumberger. Since the entrance threshold for converting MATLAB code to efficient C++ with *m2cpp* is quite low, we expect the tool to be adopted also by several other engineering groups in Schlumberger.

7.2 UC12.2: Video surveillance for critical infrastructure

In UC12.2 it was possible to compare and work on different strategies to implement algorithms. The different methodologies and used hardware platforms have unique strengths and weaknesses.

1.) Pure Workstation based architectures:
   + easy to setup
   + fast to implement algorithms with C Code
   - expensive when a standard Video Camera should be connected
2.) Pure FPGA based architectures:
   - long engineering cycle
   - usually not enough memory for a frame buffer, and not enough hardware resources to do all needed data processing concurrently, unless high end (and expensive) hardware is used
3.) Using an embedded system board with Processor and OpenCV
   + brought the fastest implementation cycle
   - the system performance and latency was not sufficient for latency critical applications
4.) Best results has been achieved with a SystemOnFPGA that uses parallel data processing on the FPGA in combination with an CPU core on the FPGA. This platform was provided by WP4.
   + quick changes on the algorithmic part,
   + short latency, and
   + acceptable scalability of the recognized objects.

4.) describes EMC2 results that allow us to quickly test and implement parallel algorithms. We can now develop applications for smart cameras in a short time and at affordable costs. By reusing the WP4 platform, we expect that the development cycle for new applications can be reduced by 80%. The embedded approach will also enable us to build smart vision systems for less cost in non-commercial applications, and help to address power consumption issues.

7.3 UC12.3: Medical imaging

EMC2 results prove that with state-of-the-art hardware, viewing can be implemented on the same hardware platform as the host process. This will enable further optimisation of clinical processes. Further, it was shown that reconstruction can be integrated on this platform, for low- and mid-end systems. High-end systems will follow when standard hardware with more cores will become available. Integrating the real-time scan process with the non-real-time process has been found to be more difficult than anticipated, and new requirements for both hypervisor technology and the real-time operating system have been harvested.

7.4 UC12.4: Control applications for critical infrastructure

The project demonstrated that open tool-chains are one of the answers towards increasing efficiency and quality of processes and products. The prototype executed during the project provided almost full functionality of the envisaged tool-chain, and was run in real-life contexts. A next step from here is the expansion of the set-up towards a larger group of users, together with a longer exposure of the designers to such environments.

7.5 UC12.5: Railway applications

In this project we were able to show the feasibility of using multicore in a fault tolerance platform which provides a generic environment suitable for railway applications, called TAS Control Platform. With the move to multi-core the existing mechanisms required extensions and adaptations. First, it had to be ensured that the safety concepts of applications based on TAS Control Platform were still valid in the new environment and secondly, a new approach was needed to leverage the capabilities of multi-core CPUs.
Both goals were successfully tackled through:

- an extension of the existing safety mechanisms for multi-core, with a focus on health monitoring,
- a new programming model and corresponding architecture to provide applications with more computational power, and finally
- the use of hypervisor technology for integrating several independent applications on the same hardware board.

Additionally, recent considerations and efforts in standardization are explained how to achieve not only safe, but also secure applications, which is becoming a demanding challenge not only in industrial automation and automotive, but has to be tackled also in future railway applications.

8. References

[1] Diez, Almudena, "D12.1 – Requirements, specifications, and evaluation plan", EMC² deliverable D12.1, November 28, 2014.

[2] Jurzykowski, Michal, "D12.2 – Preliminary designs and first evaluation of existing technologies", EMC2 deliverable D12.2, April 8, 2015.

[3] Dahle, Hans Petter, "D12.3 – First prototypes using a combination of existing and new technologies", EMC² deliverable D12.3, November 11, 2015.

[4] Salecker, Jürgen, "D12.4 – Evaluation of first prototypes", EMC² deliverable D12.4, April 12, 2016.

[5] van Helvoort, Mark, "D12.5 – Final prototypes and a description of the intended innovation", EMC2 deliverable D12.5, December 19, 2016.

[6] Antlr – Another Tool for Language Recognition. http://www.antlr.org/. Last visited 2016-04-04.

[7] L. Torczon and K. Cooper. Engineering A Compiler. Morgan Kaufmann Publishers Inc., 2nd edition, 2011.

[8] Moerland, A., L. Geerts, H. Hoogduin, L. Moore, *Smart DTI fiber tracking through fully automated placement of regions of interest*, OHBM, 2013 (Artemis - High Profile result).

[9] M. Olszewski, J. Ansel, and S. Amarasinghe, "Kendo: Efficient deterministic multithreading in software," in Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XIV, (New York, NY, USA), pp. 97–108, ACM, 2009.

[10] Carl Hewitt and Henry G Baker Jr. The incremental garbage collection of processes. 1977.

9. Abbreviations

Abbreviation	Meaning
2oo3	2-out-of-3: A fault tolerant Triple Modular Redundancy System
ANTLR	Another Tool For Language Recognition
ART	Android Runtime
ASIC	Application-Specific Inregrated Circuit
AVX	Advanced Vector Extensions to the x86 instruction set
CHR	Combined Host Reconstruction
COTS	Components off the Shelf
DAS	Data Acquisition System
DDR	Double Data Rate
DUI	Delegated User Interface
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
HDL	Hardware Description Language
HDR	High Dynamic Range
HLS	High-level synthesis
IP	Intellectual Property
LBP	Local Binary Pattern
LDR	Low Dynamic Range
m2cpp	Matlab to C++, code generator developed by W12 task 1
MATLAB	Matrix Laboratory – a high-level technical computing language and interactive environment
MRI	Magnetic Resonance Imaging
NDK	The Native Development Kit (NDK) is a is a toolset that lets you implement parts of your app using native-code languages such as C and C++
OpenMP	Open Multi-Processing, an (API) that supports multi-platform shared memory multi-processing programming
OS	Operating System
OSLC	Open Services for Lifecycle Cooperation
QoS	Quality of Service
SDM	Sparsely Deterministic Memory
SIMD	Single Instructions, Multiple Data
SoC	System on Chip
TA	Tool adaptor
TAS (platform)	Transport Automation System (platform)
TBB	Threading Building Blocks, for writing parallel C++ programs that take full advantage of multi-core performance
VHDL	VHSIC Hardware Description Language
VM	Virtual Machine

Table 9: Abbreviations