

Project Number: FP7-611404

D6.1 – Case studies specification

Authors¹

A. Grasset (Thales), T. Seceleanu (ABB), M. Torrents (UPC), R. Canal (UPC), A. González (UPC), A. Chatzidimitriou (UoA), D. Gizopoulos (UoA), S. Di Carlo (POLITO)

Version 1.6 – 27/11/2015

Lead contractor: Thales
Contact person: Arnaud Grasset Thales Research & Technology 1, avenue Augustin Fresnel 91767 Palaiseau Cedex, France Tel. +33 (0)1 69 41 60 55 Fax. +33 (0)1 69 41 60 01 E-mail: arnaud.grasset@thalesgroup.com
Involved Partners²: THALES, ABB, UPC, UoA, POLITO
Work package: WP6
Affected tasks: T6.1

Nature of deliverable³	R	P	D	O
Dissemination level⁴	PU ⁵	PP	RE	CO

¹ Authors listed here only identify persons that contributed to the writing of the document.

² List of partners that contributed to the activities described in this deliverable.

³ R: Report, P: Prototype, D: Demonstrator, O: Other

COPYRIGHT

© COPYRIGHT CLERECO Consortium consisting of:

- Politecnico di Torino (Italy) – Short name: POLITO
- National and Kapodistrian University of Athens (Greece) - Short name: UoA
- Centre National de la Recherche Scientifique - Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (France) - Short name: CNRS
- Intel Corporation Iberia S.A. (Spain) - Short name: INTEL
- Thales SA (France) - Short name: THALES
- Yogitech s.p.a. (Italy) - Short name: YOGITECH
- ABB (Norway and Sweden) - Short name: ABB
- Universitat Politècnica de Catalunya: UPC

CONFIDENTIALITY NOTE

THIS DOCUMENT MAY NOT BE COPIED, REPRODUCED, OR MODIFIED IN WHOLE OR IN PART FOR ANY PURPOSE WITHOUT WRITTEN PERMISSION FROM THE CLERECO CONSORTIUM. IN ADDITION TO SUCH WRITTEN PERMISSION TO COPY, REPRODUCE, OR MODIFY THIS DOCUMENT IN WHOLE OR PART, AN ACKNOWLEDGMENT OF THE AUTHORS OF THE DOCUMENT AND ALL APPLICABLE PORTIONS OF THE COPYRIGHT NOTICE MUST BE CLEARLY REFERENCED

ALL RIGHTS RESERVED.

⁴ **PU**: public, **PP**: Restricted to other programme participants (including the commission services), **RE** Restricted to a group specified by the consortium (including the Commission services), **CO** Confidential, only for members of the consortium (Including the Commission services)

⁵ This deliverable contains two confidential sections regarding the Thales flight management system. We added these sections to give more information to reviewers. However they will be removed from the public version posted on the website.

INDEX

COPYRIGHT	2
INDEX	3
Scope of the document	4
1. Introduction	5
2. HPC case study specification	6
2.1. Overview of the case study	6
2.1. The Sierpinski Framework	8
2.2. Approach	9
3. Industrial case study specification	11
3.1. Overview of the case study	11
3.2. Use Case Description	12
3.2.1. Drive Controller	13
3.2.2. Safety Option.....	14
3.2.3. Other design elements	14
3.3. Approach	15
3.4. Summary	16
4. Avionics case study specification	17
4.1. Overview of the case study	17
4.2. Description of the case study (confidential)	18
4.3. Task-level description & requirements (confidential)	18
4.4. Hardware demonstration platform	18
4.5. Reliability requirements and failure modes	19
4.6. Case study input data set	20
5. Conclusion	21
6. Acronyms	22
7. References	23

Scope of the document

This document is an outcome of the task T6.1 “**Case studies specifications**” described in the description of work (DoW) of CLERECO project under Work Package 6 (WP6). Figure 1 depicts graphically the goal of this deliverable, its main results, and which work packages will use its outputs.

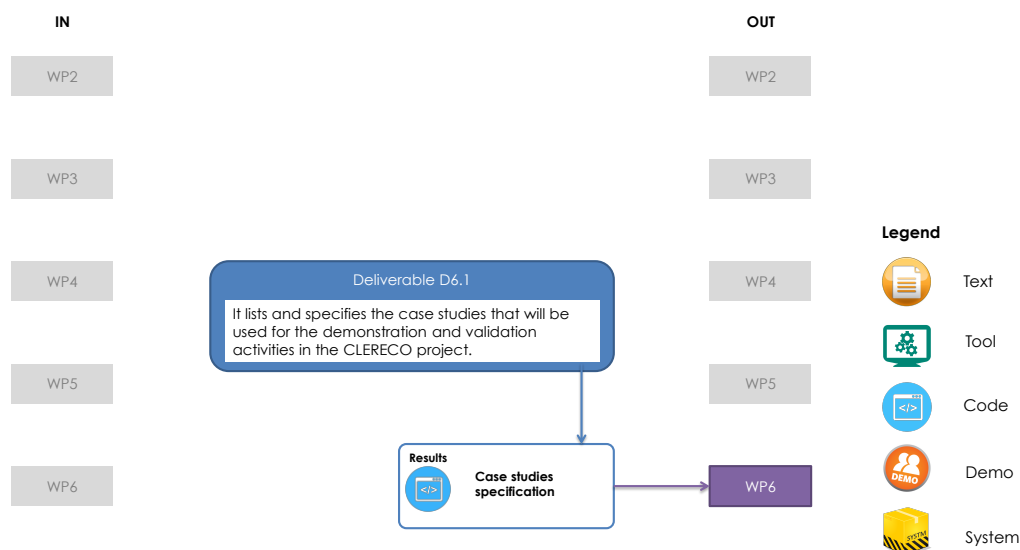


Figure 1 - Task and deliverable overview

This document contains a description of the test cases that will be used to validate CLERECO results. Three case studies have been selected and identified as relevant for the needs of the demonstration activities. They come from a wide range of application domains including High Performance Computing applications, industrial application domain as well as safety critical embedded systems. This diversity will enable us to validate that the CLERECO tools can be efficiently used for the design of systems with different kind of requirements and operating conditions. The outcome of this deliverable is thus a set of case studies that have been selected for the demonstration and validation activities of the CLERECO project.

The document is organized in the following Sections:

- **Introduction.** It introduces the role of the case studies in the project.
- **HPC case study specification.** It presents the specification of the HPC case study that consists of a full-fledged tsunami simulator.
- **Industrial automation case study specification.** It presents the specification of the industrial case study that consists of a motor controller.
- **Avionics case study specification.** It presents the specification of the avionics case study that consists of a Flight Management System.
- **Conclusion.** A short summary of the activities described in this deliverable.

1. Introduction

In the context of the decreasing reliability of semiconductor devices due to the technology scaling, evaluating the reliability of a system becomes a critical and challenging task. To deal with this evolution of technologies, the CLERECO project develops tools to perform system reliability estimation early in its design phase. However, exploitation of these tools in real projects will be possible only if developers are confident on the validity and on the accuracy of the results. Therefore, validation of the results is an important and critical task in the CLERECO project to enable the use of the CLERECO methodology and tools in an industrial context.

To this end, the use of industrial case studies is mandatory in order to have complex applications and not just simple benchmarks. The vulnerability of an application depends on its characteristics and complexity. Working with complex applications will enable to deeply understand the potential and capability of the developed methodology. For instance, a small benchmark whose code could fit in the instruction cache of the processor would not test appropriately the methodology. It is also necessary to have complex applications in order to check how the analysis effort scales with the complexity of the application.

The validation and demonstration activities of the CLERECO project rely on three case studies. These case studies are representative of different application domains, namely: the High Performance Computing domain, the industrial domain and the avionics domains. So, they cover different segments of the Computing Continuum. As the CLERECO project aims to deliver tools targeting many application domains, it is important to validate that the tools work with different case studies. One of the goals of CLERECO is to provide designers a facility to estimate system reliability for a wide range of applications.

The first case study will analyze the capabilities of our proposed methodology for HPC systems. Due to high cost of building a dedicated HPC system for our purposes, we will build a scaled-down HPC system consisting of 2 AMD Opteron Processors 6338P. After having considered several applications from the HPC domain and a preliminary analysis, a tsunami simulation application has been selected to be used for the CLERECO validation activity.

A second case study is an industrial application in the framework of motor controller. Such systems are used to control electrical motors. Their reliability is therefore mandatory to guarantee safety in industrial plants. It is a bare metal application (i.e., no Operating System). It will be implemented on a Xilinx Zynq platform based on an ARM Cortex-A9 subsystem associated to programmable logic (i.e., FPGA). The application is created starting from a high-level model that enables to implement portions of the application as software (C code) and portions as hardware (VHDL blocks).

The third case study is an industrial application use case based on a Flight Management System. This function is a fundamental part of a modern aircraft's avionics that automates a wide variety of in-flight tasks, such as localization, flight planning, guidance, predictions and so on, reducing the workload on the flight crew to the point that modern aircrafts no longer carry flight engineers or navigators. The application is planned to be executed on a ARM Cortex-A15 subsystem. Differently from the previous case study, this application is executed on top of an Operating System with no HW acceleration.

It is important to highlight here that the three use cases reflect applications from very different domains. Motivations, requirements, design styles and tools, constraints are in general very different. This is reflected in the descriptions provided in this document.

This document is the outcome of the task T6.1 where the three case studies have been studied. It is a specification of these case studies that will be used in the tasks T6.2 and T6.3 for the analysis and validation. The document is organized as follows. Section 2 presents the High Performance Computing case study. Section 3 describes the industrial case study that consists of

a motor controller. Section 4 of the deliverable is focusing on the avionic test-case expressing the general task flow of the Flight Management System use-case, and summarizing all the use-case related requirements.

2. HPC case study specification

2.1. Overview of the case study

The case study chosen by partner UPC to demonstrate the application of CLERECO reliability analysis tools in the design of a HPC system is based on a high-performance platform with several threads running a tsunami simulation framework.

In order to design an HPC case study we needed two main resources: A HPC platform and a software application able to exploit as much as possible the resources of this machine. In the following paragraphs, we will explain the most relevant details of each of these parts.

HPC hardware platforms are complex machines with high costs and only few leader providers in the world. The design of a full scale HPC system is high demanding in terms of complexity, time and resources that could not be afforded in the time frame of a single EU project. A complex activity has been therefore carried out by partner UPC in the framework of the CLERECO project to identify the specification of a system including all computing characteristics of a full-scale systems but keeping at the same time complexity, design time and costs within the available budget.

Joshua (Figure 2) is the name of the hardware platform that the UPC partner has bought for the implementation of the hardware infrastructure of the HPC case study. Joshua is a platform equipped with two AMD Opteron(tm) Processor 6338P. Each of these processors is composed of 12x86 cores that can run up to 2.8 GHz each. Each core has a 1MB L2 private cache. The whole socket has a 16MB L3 shared cache. Thus, in total Joshua has 24x86 cores each one containing a 1MB L2 private cache in addition to 2 shared L3 caches of 16 MB each. The complexity of this hardware architecture goes beyond the capability of analysis of any known hardware fault-injection reliability analysis tool and therefore represents a challenge for the tool suite developed in CLERECO.



Figure 2 - The Joshua platform installed in the UPC

Together with the hardware architecture, the HPC case study requires to identify a software application able to exploit this 24-core machine. For this purpose we have chosen the Sierpinski Framework [8]. This framework is an open source software to solve hyperbolic equations on dynamically changing fully-adaptive conforming 2D triangular grids. A kernel based way to solve hyperbolic problems and to apply steering during simulation is offered. In particular, we

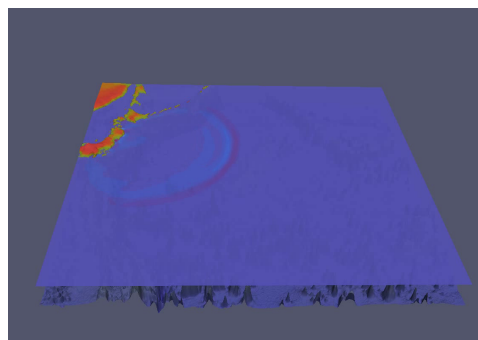


Figure 3 - A frame of the simulation of the tsunami's propagation

use this framework with the augmented Riemann flux solvers [5][6] to simulate the propagation of a tsunami over the sea as it can be seen in Figure 3. In our opinion, this application is an appropriate use case for the HPC scenario for two main reasons:

The first one is the highly computational demands this application requires and its capability to be parallelized to feed and stress all the computing cores of our system. Notice that, for each simulation step the application generates a snapshot and for recreating a 16 sec video about 400 snapshots are needed, which results in about one hour of processing time in Joshua.

The second reason is the importance of the correctness of the results of this kind of simulations. Notice that this kind of software may be used for organizations that try to predict the impact of a natural hazards in the earth, such as the Emergency Response Coordination Center (ERCC), the European Earth Observation Program, or the United Nations Office for Disaster Risk Reduction. Also in Europe, many seaside areas along the Mediterranean Sea have been affected by tsunamis. Figure 4 shows the results of a study made by the European Spatial Planning Observation Network (ESPON) in 2005 where it can be observed the potentially areas under tsunami hazard in Europe. As stated before, this kind of applications are used to predict the impact of the tsunami hazards. For this reason an error on this simulation models could be catastrophic. Imagine that the strength of the tsunami is lower than what the simulation had predicted. In this case, a city that is completely out of the scope of this tsunami could had been totally evacuated according to the prediction, taking into account the cost of this evacuation. Or even worse, imagine that the strength of the tsunami is higher than what the simulator says and the city is not evacuated just to save costs.

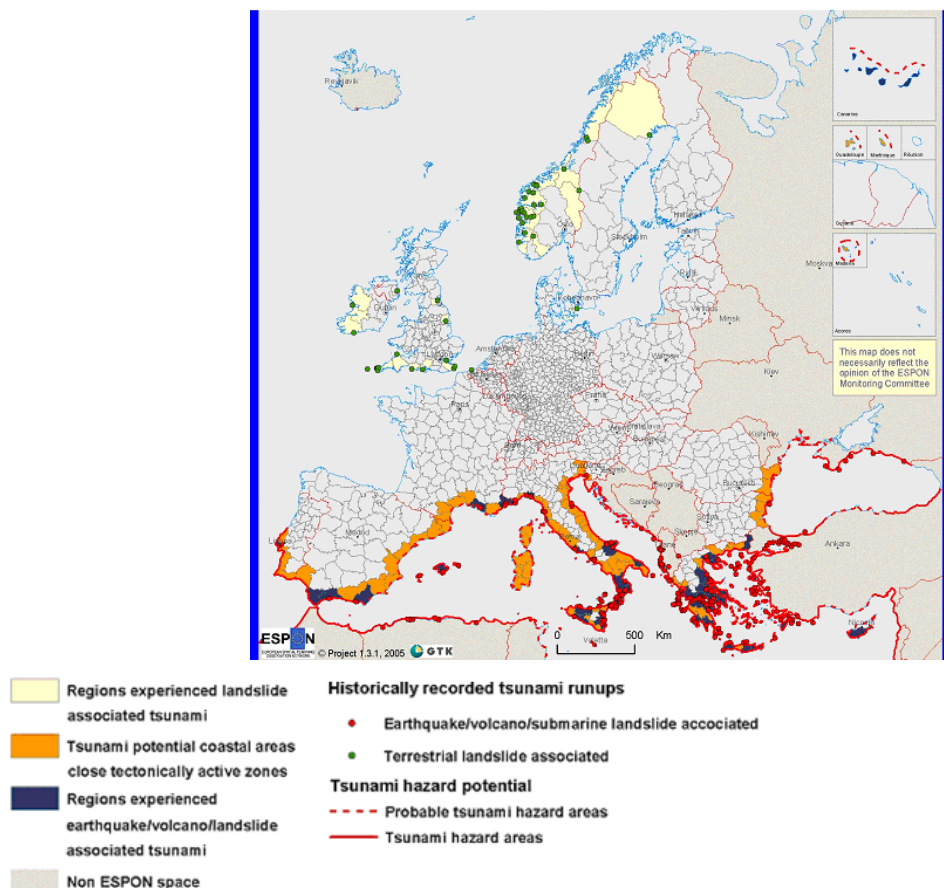


Figure 4 - European countries in tsunami hazard.

In order to have a more realistic validation process, we reproduce a real case. In particular, one of the most famous and catastrophic tsunami from the last years. It is the Tohoku Tsunami from 2011 in Japan which had catastrophic consequences, emphasized by the nuclear accidents at three reactors of the Fukushima Daiichi Nuclear Power Plant complex. The Sierpinski framework, analyzes the starting configuration of the area, and calculates the propagation of the wave for a determined number of steps.

2.1. The Sierpinski Framework

The tsunami propagation is calculated through a solver, which is a numerical method for the solution of hyperbolic partial differential equations. This solver is integrated in the Sierpinski framework and it is the responsible for accurately predicting the behavior of the tsunami wave. Moreover, the Sierpinski framework can work with several solvers [5][6]. All of them solve the hyperbolic partial differential equations but in a slightly different way. This is important from the reliability point of view because different solvers may make more or less use of arithmetic or logic operations. Thus, there can be more reliable solvers than others. In particular, the Sierpinski framework can work with the following solvers:

- Lax friedrich
- Rusanov
- Fwave
- Augmented Riemann (C++)
- Hybrid solver
- Augmented Riemann (GeoClaw)
- Augmented Riemann (SIMD)
- Velocity upwinding
- FullSWOF (external)
-

In addition to the solvers, the Sierpinski framework contains a large number of configurable parameters, which are listed in an xml file used as an input of the framework when compiling or executing. These parameters are classified into four different groups: compiler, threading, sierpi, and tsunami parameters. The most important parameters for each of these groups are briefly cited bellow:

- **Compiler:** parameters about the type of compiler (e.g., Intel or Gnu) and other aspects that must be defined at compile time. The most important of them are the threading framework and the solver.
- **Threading:** Parameters to determine the number of threads and their behavior during the simulation.
- **Sierpi:** Parameters to determine the behavior of the Sierpinski framework, such as the verbosity level of the execution, simulation time, or the frequency of data output.
- **Tsunami:** Parameters related to the input grid, the relative recursion depth of the simulation and the size of the grids or constant values such as the gravity.

As shown in Figure 5, the framework uses the bathymetry files as part of the configuration file introduced above. These bathymetry files are generated by international institutions, concretely we are working with data provided by the General Bathymetric Chart of the Oceans (GEBCO) and National Oceanic and Atmospheric Administration (NOAA). For this reason, in

order to use them in the Sierpinski framework, these files must be preprocessed by a software provided by the people in TUM (Munich). This software is called Tsunami (<https://github.com/TUM-I5/tsunami>) and its purpose is to transform the bathymetry files to data manageable by the tsunami simulation software.

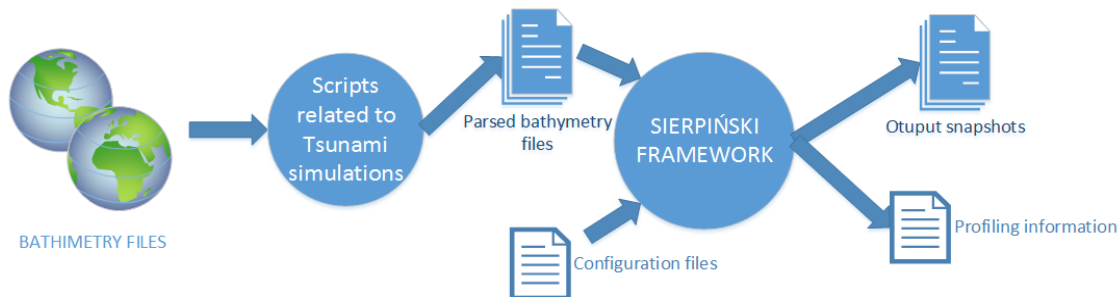


Figure 5. Source and output files related to the Sierpinski framework.

2.2. Approach

The system (hardware infrastructure and running software) will be modeled resorting to the CLERECO reliability analysis tools in order to perform the estimation of the reliability of this HPC system. The challenge of this use case in our project will be the ability to manage the complexity of the hardware platform and the complexity of the software application in the very short time frame available for the project in order to obtain reliability estimation to be compared with those provided by commercially available tools.

Succeeding in the analysis of this use case will be a clear demonstration of the capability of our tool to scale with complex applications and to enable to obtain fast results even when the complexity of the system to analyze is high. Moreover, the availability of different solvers in the selected application can be used to exploit the reliability reasoning capability of the CLERECO tool to identify those solvers that enable to reach a better reliability.

In order to test the framework we have prepared three configuration files. The main difference between them is the relative recursion depth of the simulation, which is directly related to the number of points from the resulting result of the simulation. **Table 1** shows the main differences between the input sets: large, medium and small. The differences can be observed in the “sim time” (in Joshua machine), the number of processed cells, and a sample image of the result. Nevertheless, the large input is the one that is going to be used for the validation process.

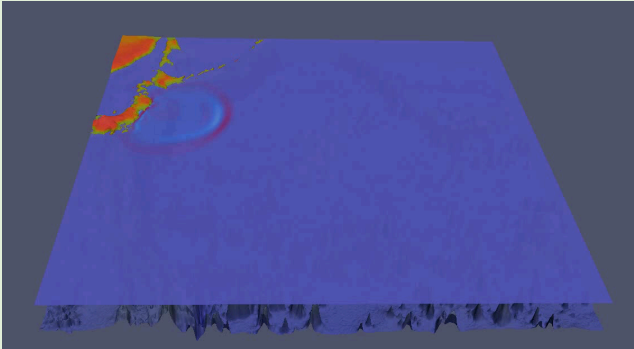
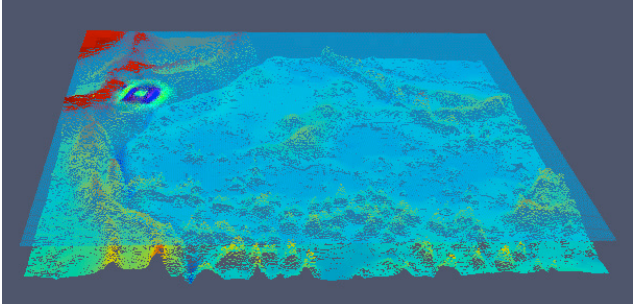
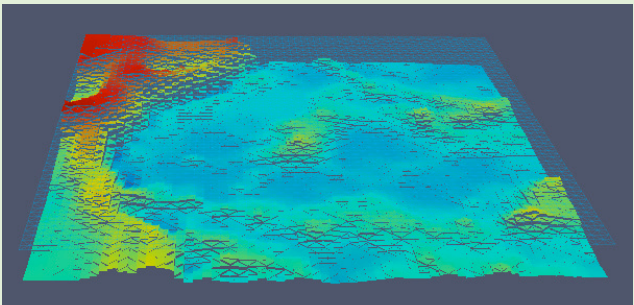
Input set	Sim time	Processed Cells	Sample
large	1360 s	10720 Million	
medium	47s	163 Million	
small	2s	2 Million	

Table 1. Input sets of the validation framework.

3. Industrial case study specification

3.1. Overview of the case study

The case study chosen by ABB to demonstrate the results of the project is centered on the realization of a motor drive (controller).

A motor control system is a very common application in the industrial domain. It can be used in a simple conveyor belt with one rotating motor or a long belt with carefully synchronized multiple motors to minimize forces on the belt. It can be exploited in an indoor protected area like a paper mill or out in a ruggedized environment transporting stones and cement.

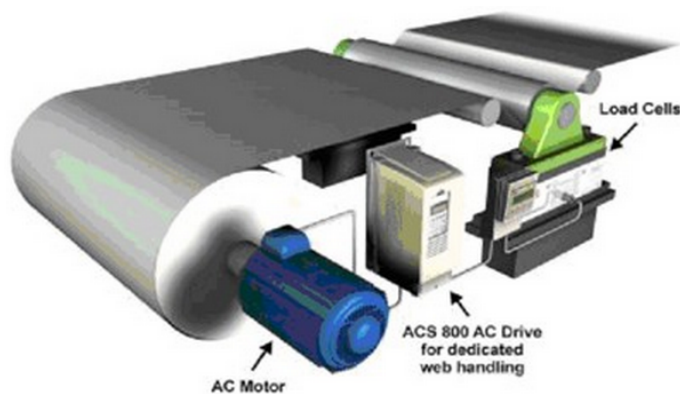


Figure 6 - Conveyor belt with drives

It can also be exploited in a complex process environment consisting of fans, pumps, compressors, etc. The functionality can be an advanced factory automation utility consisting of robots, machines, and humans in constant interaction.



Figure 7 - Robot with control system

3.2. Use Case Description

The contexts in which motors described above operate are more often than never imposing certain levels of safety. The implementation of safety related regulations comes usually as additional modules and functions within the motor controllers.

A typical safety application for rotating motors is an emergency stop button. This is because the human being often is close to the motor. If something happens, there must be a possibility to immediate turn the motor off without questions asked. The motor sizes can range from tiny non-cost to huge expensive. The motor controller then also serves as a protection device for the motor.

Figure 8 represents a typical and simplified ABB Motor Controller, consisting of the Drive Controller (DC) and the safety option (within the red square). The Control System (CS) is represented by the HMI (Human Machine Interface) and Configuration on top.

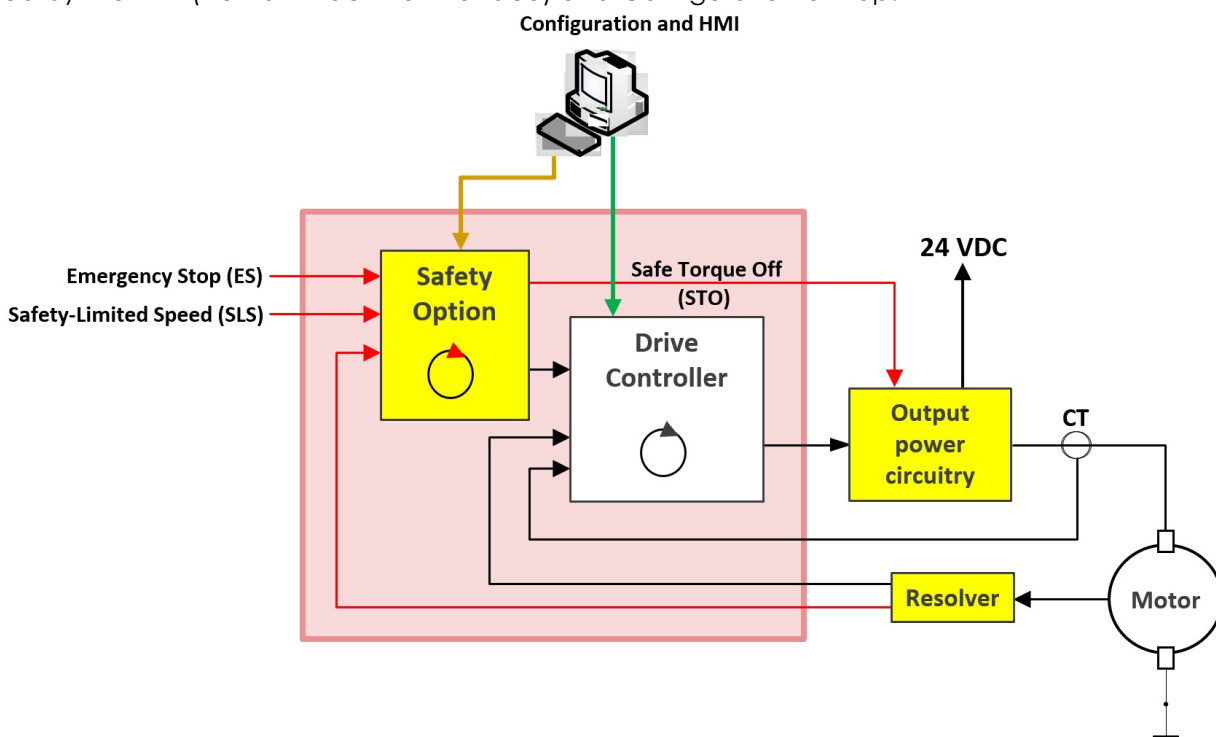


Figure 8 - Motor Controller, closed loop

Safety critical components are marked in yellow.

Within CLERECO we focus on two safety features: Safe Torque Off (STO) and Safety Limited Speed (SLS).

Safely-limited speed (Figure 9) ensures that the specified speed limit of the motor is not exceeded. This allows machine interaction to be performed at a slow speed without stopping the drive. The safety functions module comes with four individual SLS settings for speed monitoring. The implementation should provide the possibility to change the speed limit of the SLS on the fly.

Safe torque off (Figure 10), when activated, it brings the machine safely into a non-torque state and/or prevents it from starting accidentally. The function can be used to effectively implement the prevention of unexpected startup functionality, thus making stops safe by preventing the power only to the motor, while still maintaining power to the main drive control cir-

cuits. Prevention of unexpected startup requires for example a lockable switch in addition to the STO function. Note that STO does not protect against electrical hazards.

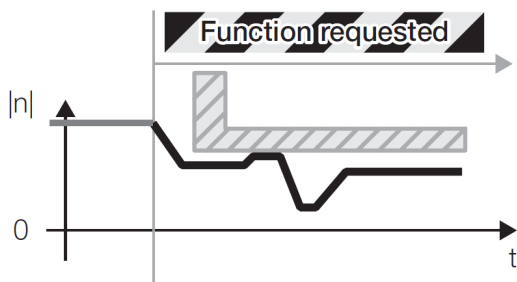


Figure 9 - SLS behavior.

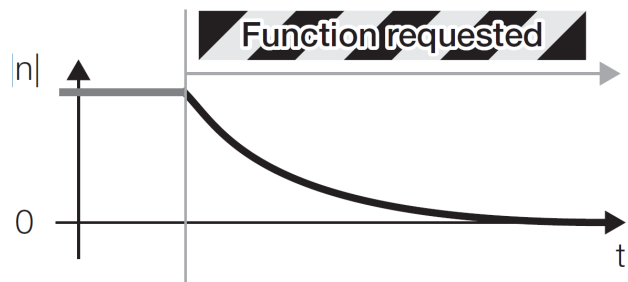


Figure 10 - STO behavior.

Usually, the implementation of STO and SLS is localized in a physically separate module (depending on the requirements). These modules may be dedicated to only one safety function – see Figure 11, or to several functions – see Figure 12.



Figure 11 - An ABB module containing Safe Torque Off Function



Figure 12 - ABB safety functions module (SLS among them).

However, in CLERECO, we will use a different platform for the implementation of such functionalities (together), in order to accommodate multiple possible variations of both functionality and targeted technology.

3.2.1. Drive Controller

The Drive Controller is continuously instructing one or several motors what to do. Set points are typically given via the HMI and this is transformed into motor commands. DC continuously receives feedback of position (and speed) from the resolver. This is used for running the motor control loop.

- HMI gives commands to the DC, which is translated into motor commands.
- The resolver continuously measures motor position and feed this back to the control loop for supervision.
- The current transformer (CT) protects the motor from overcurrent conditions comparing calculated and actual values.

- In addition, it reacts on commands from the Safety Option (Emergency Stop and Safety Limited Speed).

Implementation details around the DC to be established:

- Select HW (standard test PCB with relevant CPU)
- Select operating system, if any
- Develop application for motor speed control, motor overload protection, HMI interaction, safety option interaction
- Digital output for PWM control (Pulse Width Modulation)
- Analog inputs to handle resolver signal and current measurements
- Communication solution towards safety option

3.2.2. Safety Option

This Safety Option is the motor safety supervisor. It continuously monitors the motor activities via the resolver feedback. Based on predefined motor limitations, it can cut of the power to the motor. This normally means the DC has failed or is not aligned with the SO.

- SO is connected to HMI or Control System (CS) to receive motor configuration data (motor characteristic).
- SO takes also direct input from user via ES and directly disables the power to the motor in addition to informing the DC.
- It receives Resolver feedback making it possible to supervise DC – not based on motor commands, but on motor characteristic (overload, overheating, etc.).

A simplified SO solution could be not to supervise DC, but only adding an Emergency Stop button.

Implementation details around the SO:

- Select HW (standard test PCB with relevant CPU)
- Select operating system
- Develop application for STO, SLP, HMI interaction, drive controller interaction
- Output power circuitry
- 2 digital inputs for STO and SLS
- 1 digital output for STO enforcement
- Analog input to handle resolver signal
- Communication solution towards drive controller

3.2.3. Other design elements

A Resolver is an ultra-reliable rotary motion-sensing transformer mounted on the rotating shaft of the motor to detect all motor movements. It reports the motor position and speed to the Motor Controller system closing the control loop.

Configuration and HMI is representing the Control System for configuring and setting up the motor controller with the correct motor characteristics – typically threshold values and criticality – used for selecting application configuration and all required user interaction.

Safe Torque Off ensures that there is generated no motor torque due to energy from the DC and Output power circuitry. This is enabled by pressing the Emergency stop (ES) button or when the Safety-Limited Speed is exceeded. When activated, the STO function ensures that there is generated no motor torque due to energy from the DC and output circuitry.

Implementation details around the DC: utilize standard resolver and current transducer suitable for the application.

3.3. Approach

Several scenarios for the implementation can be taken into account as follows.

1. No safety: the SO can be removed, we only implement the DC.
2. Limited safety: The SO and DC can be embedded on the same platform, as different software applications running different processes.
3. Full safety: The SO and the DC can be implemented on separate platforms consisting of different hardware (HW) and software (SW).
4. Full safety with redundancy: The SO and the DC can be implemented on separate platforms consisting of different hardware (HW) and software (SW) with redundant SO solution.

Of the above options, only the last one seems outside of the project reach – due to the relatively limited number of resources.

Otherwise, in addition to the scenarios, further variations can be obtained in the realization of the use case, in order to be able to observe an as wide as possible impact of potential reliability failures, and to provide solutions for addressing them. In this sense, the expectation is that starting with a 100% software implementation elements of design would be gradually moved into hardware as potential high impact errors may be prognosticated with high probability.

Platform. In order to support the above goals, a Xilinx Zynq platform has been selected to implement the use case. It consists of a processor connected to an FPGA logic area via AXI interfaces. It gives the possibility to implement both hardware and software modules of the demonstrator. It further gives the possibility to quickly move such models from one technology to the other (at design time), following potential negative impact results from the analysis performed in the project.

The design will consider bare-metal implementation for the software part, that is, no operating system to run on the platform's processors.

Process. Further, our approach to the use case is organized in two stages.

A generic model at first will be used to study both the potential of the tools and the quality of the implementation and of the selected platform. High level specifications of safety options are to be implemented. Following a characterization of the implementation with respect to reliability aspects, a set of "lessons" are to be extracted and used in the second stage of the developments.

The second phase of the approach targets the implementation of a refined, more specific design, with more accurate safety specifications to be implemented. The resulting system is planned to be demonstrated as operating a suitable size (dimensions and cost) motor. Final project outcomes are to be tested on this design.

The tool landscape. Tools in the process should support the above description of goals. The Mathworks' Matlab-Simulink environment has been chosen for modelling the motor controller. There are many generic examples of a Motor Control configuration in Simulink that can be used at least in the first stage of the approach, for generating C-code and VHDL to be implemented on the selected target system. The same tool provides also means to automatically generate both C code for the software parts, as well as VHDL for the hardware parts. Hence, a quick change in the targeted implementation technologies should be possible.

For the hardware part, the Xilinx Vivado environment has been selected. It will both synthesize the VHDL elements and support the integration of software and hardware modules.

3.4. Summary

A quick overview of the choices and approach in the realization of the use case is presented below.

	Stage 1	Stage 2
Platform	Zynq	
Tools	Mathworks	
	Vivado	
Result	3 scenarios	
	Hardware / software "mobility"	
	Generic models	Specific models
	Simplified safety options	Final safety options
	Motor models	Physical motor demonstration
Timeline	M26	M34

4. Avionics case study specification

4.1. Overview of the case study

One of the case studies selected by Thales to be representative of safety-critical applications is a Flight Management System (FMS)[1]. It is an industrial avionics application whose purpose in modern avionics is to provide the crew with centralized control for the aircraft navigation sensors, computer based flight planning, fuel management, radio navigation management, and geographical situation information (Figure 13). Taking charge of a wide variety of in-flight tasks, the FMS allows us to reduce the workload of the flight crew allowing us to reduce the crew size. The FMS is especially responsible for the flight management services allowing in-flight guidance of the plane. From pre-set flight plans (take-off airport to landing airport), the FMS is responsible for the plane localization, the trajectory computation allowing the plane to follow the flight plan, and the reaction to pilot directives (see Figure 14).



Figure 13 - Illustration of a Flight Management system

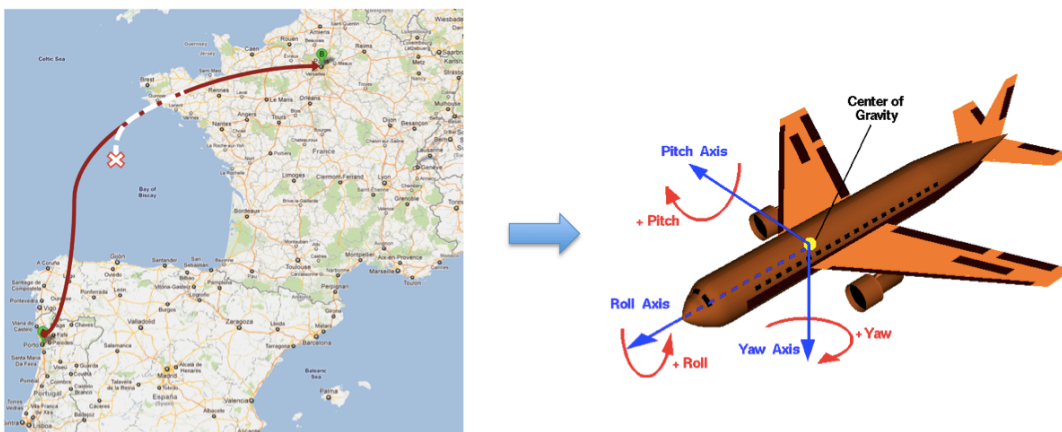


Figure 14 - Flight Management System (FMS)

The Flight Management System is an industry level application characterized as real-time application with different kinds of hard real-time requirements, and with high reliability and high availability requirements. As a safety-critical application, the FMS is situated at one extremity of the computing continuum (see Figure 15).



Figure 15 - FMS Case Study positioning in the computing continuum

The application is based on a commercial Flight Management System but it has been rewritten in a multi-threaded way. So, the application can be executed on a multi-core processor and can be used to study the propagation of faults in such processor architecture. Even if it is a simplified version of a Flight Management System, the application is still representative of avionics application in terms of performance requirements and in term of memory requirements.

4.2. Description of the case study (confidential)

4.3. Task-level description & requirements (confidential)

4.4. Hardware demonstration platform

The demonstration platform that has been selected for this case study is based on a 66AK2E05 multicore KeyStone II System-on-Chip from Texas Instrument [2]. Its main features are:

- Four ARM Cortex-A15 Processor Cores at up to 1.4-GHz
- 4MB L2 Cache Memory Shared by all Cortex-A15 Processor Cores
- 32KB L1 Instruction and Data Caches per Core
- One TMS320C66x DSP Core Subsystem
- 2MB of Multicore Shared Memory (MSMC) that can be used as a shared L3 SRAM

Figure 16 shows the block diagram of the Keystone II SoC.

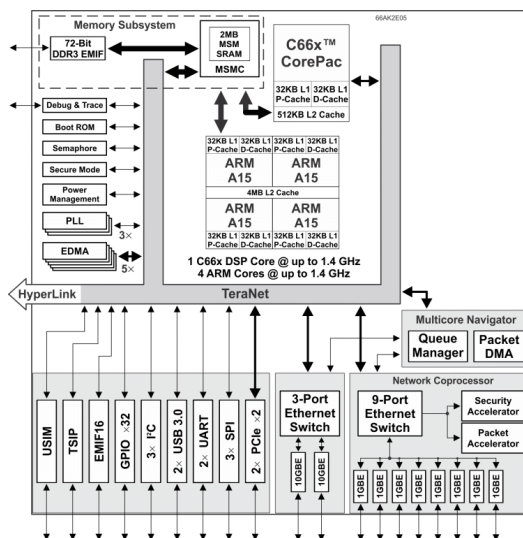


Figure 16 - Block diagram of 66AK2E05

The board used for the development and for the demonstration activities is the EVMK2E Evaluation Module provided by Texas Instrument [2].

The selection of this platform is motivated by many reasons:

- the ARM Cortex-A15 processor model is available in the WP3 workpackage
- the ARM architecture is massively used in the embedded domain
- the Keystone-II is an advanced and complex Multi-Processor System-on-Chip

An embedded Linux OS will be used to ease the SW development and to be compatible with the analysis done in WP4. Avionic application like the FMS are generally executed on a

RTOS, so it is important for this case study to validate that CLERECO methodology can be used to analyze the whole SW stack. If using a RTOS is important in an operational context due to the real-time requirements of the applications, using an embedded Linux is sufficient in the CLERECO context regarding the goals that we want to achieve. We can assume that if the CLERECO tools can support a Linux OS, they could support a light RTOS.

4.5. Reliability requirements and failure modes

The aerospace safety process is a top-down process driven from the safety requirements. So, the reliability requirements of an application derive from the safety analysis. A Design Assurance Level (DAL) is assigned to each function of the system according to the severity of its potential failure. Acceptable failure rates are associated to each DAL (see Table 2). The FMS being responsible of the guidance of the aircraft, it is a critical application whose safety level is set to DAL B.

Level	Severity	Consequences	Failure rate (per hour of flight)
A	Catastrophic	Failure may cause a crash. Error or loss of critical function required to safely fly and land aircraft.	$< 10^{-9}$
B	Hazardous	Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers.	$10^{-7} \rightarrow 10^{-9}$
C	Major	Failure is significant, but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries) or significantly increases crew workload.	$10^{-5} \rightarrow 10^{-7}$
D	Minor	Failure is noticeable, but has a lesser impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change)	$10^{-3} \rightarrow 10^{-5}$
E	No Effect	Failure has no impact on safety, aircraft operation, or crew workload.	$>10^{-3}$

Table 2 - Safety level definition

As the application is safety critical, every deviation of its outputs should be considered as a failure of the system (whatever is the consequence at the aircraft level). At a coarse-grain level, we can distinguish outputs to the display (localization, flightplan trajectory and nearest airport list) and to the auto-pilot (guidance information). Moreover, latent faults have also to be considered. For instance, a corruption of a database that would not have caused a failure during the experiment should be detected as it could potentially impact the next flight.

Any violation of the real-time requirements should also be considered as a failure.

4.6. Case study input data set

The workload used in the case study corresponds to the emulation of a regional flight between two airports with a small number of waypoints. In the context of this case study, the programming of the flight plan has been embedded directly in the code of the application in order that the application does not have to handle the input commands coming from a Control Display Unit. An asynchronous task emulates the actions of the pilot at the beginning of the execution. Figure 17 shows the flight plan that is currently pre-programmed in the application.



Figure 17 – Flight plan used as input dataset

In order to have workload of different durations, different flight plans will be pre-programmed in the application. The selection will be done through the selection of different compilation options. Considering that the duration of a small flight is generally of at least of one hour, emulating a realistic flight is clearly incompatible with the requirement to be able to execute the application a very high number of times for fault injection campaigns. The execution of the workload is thus reduced by use of an extremely high (and unrealistic) aircraft speed as a parameter of the application.

In addition, the sensors input in the application have been replaced by an internal feedback loop. This simple solution avoids the complexity of emulating the sensors and simulating the behavior of the aircraft. The output of the trajectory task that corresponds to the future position of the aircraft that is targeted is directly transmitted to the sensors task that uses it as the position of the aircraft. This configuration of the application does not require managing real-time inputs and greatly simplifies the experimental set-up.

5. Conclusion

This deliverable presented the specification of the three case studies that have been selected from different application domains. The first case study is representative of HPC systems. It consists of a scaled-down HPC system based on 2 AMD Opteron processors (with 12 CPU per processor) and of a tsunami simulation application. It is a real HPC application that requires high computing power and long execution times. The second case study comes from the industrial application domain. It is a motor controller application implemented on a Xilinx Zynq platform. Different HW/SW trade-offs are possible for this application. The design space exploration is thus one of the challenges of this application. The third case study is based on a Flight Management System. It is an avionic application with high safety requirements. It is however one of the most computation intensive application in the avionic domain, which is developed to be run on an Operating System.

Three case studies representative of industrial and complex applications have thus been selected. It will enable to validate that the CLERECO methodology is applicable to complex application without requiring prohibitive analysis efforts. Moreover, these applications come from a wide range of applications domains. It is an important point as the aim of CLERECO is not to develop a domain-specific tool but to be sufficiently flexible and generic to target a broad range of application domains. The selected case studies cover the embedded world as well as the HPC world. In the embedded systems domain, the two selected case studies have also different requirements. One is a bare-metal application with HW acceleration, while the other is OS-based with no HW acceleration.

These case studies are going to be used in the tasks T6.2 **“Use cases reliability analysis”** and T6.3 **“Validation”**. Each case study selected in Task 6.1 will be analyzed using CLERECO reliability estimation methodology and exploiting the EDA tool developed in work package WP5. The results of the task 6.2 will be carefully validated resorting to simulation-based techniques (e.g., fault injection) applied at the end of the design phase. Based on the use case, we will use different types of fault injection (high-level, low-level).

6. Acronyms

Acronym	Full text
DC	Drive Controller
CS	Control System
HMI	Human Machine Interface
STO	Safe Torque OFF
SLS	Safety Limited Speed
CT	Current Transformer
HW	Hardware
SW	Software
SO	Safety Option
PWM	Pulse Width Modulation
ES	Emergency Stop
CPU	Central Processing Unit
PCB	Printed circuit board
FPGA	Field-Programmable Gate Arrays
FMS	Flight Management System
DAL	Design Assurance Level

7. References

- [1] Randy Walter, "Flight Management Systems", in The Avionics Handbook, Cary R . Spitzer, Eds. CRC Press , 2001, ch. 15.
- [2] Texas Instrument, "66AK2E05/02 Multicore DSP+ARM KeyStone II System-on-Chip (SoC) (Rev. D)", datasheet, 11 Mar 2015.
- [3] Texas Instrument, "EVMK2E Technical Reference Manual", Version 2.0, November 2014.
- [4] M. Schreiber, H.-J. Bungartz and M. Bader, Shared Memory Parallelization of Fully-Adaptive Simulations Using a Dynamic Tree-Split and -Join Approach, IEEE Xplore, Puna, India, 2012
- [5] Solver: D. L. George, Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation, J. Comput. Phys., 2008
- [6] M. J. Berger, D. L. George, R. J. LeVeque, K. T. Mandli, The GeoClaw software for depth-averaged flows with adaptive refinement, Advances in Water Resources, 2011
- [7] PIN. http://rogue.colorado.edu/Wikipin/index.php/Main_Page.
- [8] Sierpinski Framework <http://www5.in.tum.de/sierpinski/>