Safety-Critical Platform Model Based on Certification Standards

José Luis Gutiérrez[1] , Jesper Berthing[2] , David Fernández[3] and Javier Díaz[4]

Abstract— **Nowadays, the necessity of developing safety systems has led to the investigation of new tools and methodologies to be added to the development process of elements on which human being lives rely. This goes from the improvement of the requirements traceability to the system design automation. Minimizing manual design and coding allow achieving this goal and it is possible thanks to the utilization of high level languages that allow capturing the intended behavior of the system. The final implementation can be obtained by using proper synthesis/compiling tools allowing the production of high reliability systems. In this context we present a co-design methodology based on the use of SystemC languages for the development of safety critical systems. The utilization of proper simulation tools and libraries joins onto a co-design methodology that separates the design into hardware and software elements at a later stage. This makes possible systems verification towards reliability before the final implementation process takes place, giving rise to a reduction in the implementation costs of safety systems.**

Keywords— **Certification, IEC 61508, Safety-Critical System, Safe Channel, Diagnostics, Multi-core, Redundancy.**

I. Introduction

SY *stemC*® [1] is a powerful programming language specially designed for system modelling at high level of abstraction but also capable of generating descriptions of components at Register Transfer Level (RTL) [2]. It allows moving from the highest to the lowest level of abstraction based on design specifications [3]. This description language has strongly emerged increasing its worldwide acceptance as the complexity of designs also increases, because of the possibility of developing at different abstraction level. SystemC is able to describe components at functional level and, using an iterative approach, focus on a final detailed implementation by the redefinition of these components. Thereby, SystemC could be used from the description of a functional model at a preliminary stage of development to the final process of component implementation. During next development stages, these components could be specified as hardware (HW) or software (SW) elements, applications, operating systems, or even middleware libraries that can be instantiated. These are very important features because modern embedded systems tend to use SoC as a physical platform. In particular, many reconfigurable hardware devices and tools focused on this methodology [4,5,6,7]. Since SoCs design combines hardware and software elements, the previous properties of SystemC make this language a very representative tool for embedded systems design. Moreover, there are different third-party tools, such as Catapult C [8] or Vivado [9], which can be used to synthetize RTL code directly targeted to digital hardware as FPGAs. By this, we would be able to describe at a very early stage of the development of systems the benefits that can be extracted from the SystemC description [10].

Thanks to these advantages of the development methodology already described, the SystemC programming language could be a good choice for the development of critical systems. Critical systems [11] are those in which an anomalous behavior could cause or contribute to a failure resulting in a dramatic situation within different transport vehicles, such as aircrafts, on which human being lives depend. Critical systems can be grouped by safety critical and mixed-criticality systems, depending on the grade of safety required by all the components that are included in these systems. In addition, these systems must follow up different types of certification standards in relation to the domain in which they are applicable. Some of these standards are: avionics, automotive and industry certification standards [11,12,13]. SystemC allows describing the whole SoC elements with a uniform language and using synthesis and compilation tools to automatically generate the final system implementation. The standard design flow requires that, after the system has been modelled using a high level language, the software is coded using typically C/C++ language, and the digital hardware is coded by the utilization of VHDL or Verilog. This coding process is typically done manually and therefore, it is prone to errors. Besides, working with SystemC simplifies the manual writing of code since there is just one and uniform required language, reducing human errors and improving the trazability of the requirements [2]. Both properties are essential for improving safety and design procedures of safety critical systems and, therefore, it has motivated the choice of this language for critical systems specifications.

In addition to the previous improvements, one of our objectives using SystemC as a library is to add to a design the possibility to describe mixed-criticality systems based on multicore systems using different partitioning and isolation schemas for both critical and non-critical tasks. This partitioning could be in time or in space. The fact of working with mixed systems brings the possibility of using commercially available Off-The-Shelf (COTS) for all non-critical

[1]Dpto. de Arquitectura y Tecnología de Computadores, Universidad de Granada, e-mail: jlgutierrez@atc.ugr.es.
[2]Danfoss Drives A/S, e-mail: jbe@danfoss.com.
[3]Dpto. de Arquitectura y Tecnología de Computadores, Universidad de Granada, e-mail: dfernandez@atc.ugr.es.
[4]Dpto. de Arquitectura y Tecnología de Computadores, Universidad de Granada, e-mail: jdiaz@atc.ugr.es.

functionalities of the system. One of the features of time partitioning is the possibility of using virtualization methods and tools to manage different shared resources within a platform. Elements in a system can be critical and non-critical (including cores),and these elements are part of the systems' space problem of partitioning and isolation, which is one of the key points in mixed-criticality systems.

For a valid development process in this field, the elements that form part of these systems should be described as critical and non-critical elements, and also as hardware and software, depending on the description and specification of the system. This fact brings us to the co-design concept shown in the Fig. 2. This methodology is also applicable to SystemC models because of the possibility of development at the different abstraction levels that SystemC provides.

The rest of the paper is organized as follows. Co-design methodology is described in section 2 of this document. Section 3 presents the case study chosen to illustrate the approach here presented. Section 4 presents some preliminary results and in section 5 we present the main conclusions and indications for future work.

II. SystemC co-design methodology for mixed-criticality systems

One of the goals using SystemC is the possibility to introduce safety issues and isolation methods during the co-design process at an early stage of design. Unfortunately, this kind of methodology needs a variety of accurate descriptions that are not available in many cases. This lack of detailed characterization can be extended to the COTS utilization domain. As described before, SystemC allows modelling entire systems, including software and hardware components. In spite of the big amount of features and advantages that SystemC includes in relation to time behaviour, partitioning, timing, power issues, and solutions oriented to the synthesizing of code for embedded systems, this set of libraries does not include any safety-critical API. Therefore, they can neither explicitly provide nor even test any safety features for the systems modelled by themselves.

For this reason, we have been working on the development of additional libraries and processes based on SystemC elements and components that provide models with safety features. These component libraries include the design of safety channels, runtime monitoring elements and functional requirements specification modules. This paper focuses on our development of a safety-critical platform that includes a safety channel architecture 1oo2 described on the certification standard IEC61508. There are two different kinds of models depending on whether the relevance of the model relies on the communication process among modules, or on the flow of digital signals between hardware registers and the algorithms that manage these signals: Transaction-Level Model (TLM) and Register-Transfer Level (RTL).

TLM models are closer to how SystemC modules interchange data between their interfaces/ports, relegating the processing of this data inside the modules to a secondary position. These models contribute to the testing process of different components within an architecture, providing the possibility of verifying the interoperability between different elements (using a generic input/output interface). For example, one of the advantages of using TLM instead of RTL lies in the use of a different type of busses, making tests in a same architecture search for the best timing results [3]. On the other hand, RTL models aim to signal handling and are more often used for synthesis purposes, since these can be directly injected into digital hardware.

At this point, it is necessary to go into details about the co-design methodology that involves mixed-criticality systems. Fig. 1 shows the v-graph that illustrates the development process of safety-critical embedded systems. This process allows locating the different elements of the cycle that corresponds with the methodology that is described along this document. As Fig. 2 shows below, co-design is a process in which software and hardware developments are separated at a very late stage of development into two different tasks. In the first case, requirements and specifications are taken into account to build a description of the entire system. Once the specifications have been studied, the system is split into hardware and software elements, and the development process of hardware and software will be developed separately. These two separated branches of the tree will arrange and execute tests and verification methods in order to verify if both hardware and software achieve the initial specifications, and if the interface works properly. When the first cycle of design is over, both hardware and software are joined together, tested and verified.

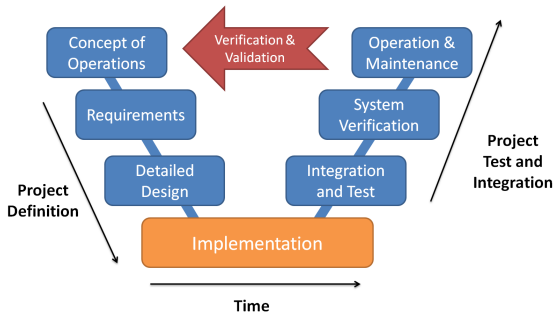

Fig. 1. V-Model. The V-model represents a software/hardware development process. The process steps are bent upwards after the coding phase to form a typical V shape. This model demonstrates the relationships between each phase of the development life cycle and its associated testing phase.

This process ends when both hardware and software fit perfectly with the specifications described at the beginning of the development process. This is an iterative process that, thanks to the utilization of high level description languages, could be significantly accelerated since the impact of moving a com-

ponent from software to hardware or vice versa could be easily quantified.

Many are the advantages of using a co-design methodology in mixed-criticality systems For example, test benches can be reused for both independent development tests (HW or SW). As these tests can also be used at the end of a design cycle, the cost of the development process decreases.

Other advantages correspond to the highest level of abstraction in the co-design development process: the system description. Since requirements are described in the first phase of design, it is possible to add traceability properties to the system description and extend them towards the end of the development process. This traceability allows the evolution of the components in different directions, providing these elements with a more concrete shape that describes its nature. For example, a shared resource can be described in the first stage of development just as a safety-critical element. In the next stage of development, the element is tested and verified but a new requirement is needed, the element should follow the automotive certification standards and its date must be duplicated. The new traceability element shares redundancy with the previous one safety-critical systems. Thus, elements obtain a more concrete shape along the co-design development process and every single step is extremely linked with the previous and the next ones.

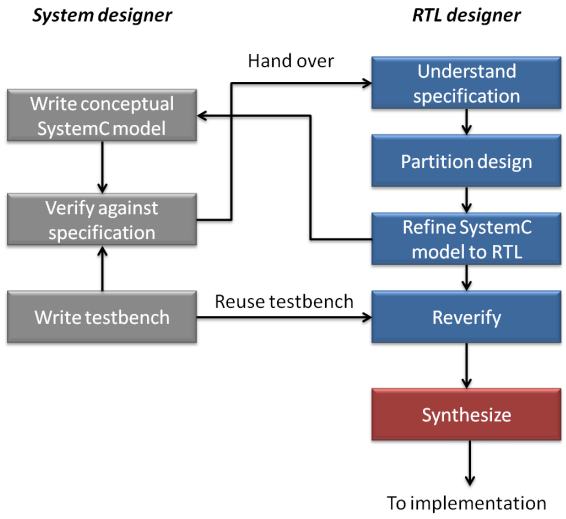

Fig. 2. SystemC Methodology (Extracted from [2]). It refers to a process for developing systems that, after describing the specifications of the entire system, is divided into software and hardware elements. Hardware and software elements are developed and tested separately till the last step, in which both elements are tested and verified with the same methods.

Co-design also supports the utilization of TLM and RTL models for hardware and software. Therefore, by using TLM and RTL models and co-design methodologies, it is possible to develop different types of elements that provide the system with safety features. The elements that we are considering on our approach are: redundancy within safe communication channels (Fig. 4), Error Detection and Correction (EDAC) modules and run-time monitoring techniques. The goal is to make possible the development of safety features that allow for isolation between critical and non critical elements of the systems. Note that although all of them are relevant and useful for embedded systems to reach a certain degree of safety, this document will focus on the development process and details of adding redundancy to a safe communication channel. This choice comes from the necessity of following the certification standards [11] during the development process of a safety platform with two critical cores in a safety critical platform.

At the time of implementing this kind of development philosophy, we have to link efforts with concepts, descriptions, safety components and target objectives on which the whole methodology can be tested. From all these necessities a powerful description method emerges: a case study. A case study is an intensive, descriptive and explanatory analysis of an individual unit that shows the elements that constitute a system and the interactions between them. We have chosen one from an industrial scene that shows the safety methods regarding the redundancy of input and output data from two cores. This case study is described along the next section: a safety platform library based on SystemC.

III. Case Study description

We have developed a SystemC model that follows the descriptions and specifications of the case study presented by Danfoss[15], used as a basic, complete and real (rich in safety-critical and certification concepts) example for the RECOMP project [16]. This case study illustrates the execution of a safety function related to the removal of a torque from an industry machine including the specification for safety-critical systems. Fig. 3 illustrates the concept that involves this case study. The safety function that is implemented by the system is to remove the torque from the motor at the moment (a short delay is allowed) an emergency stop button is pushed. There are two elements between the torque and the emergency stop button: a safety platform that is originally described as a hardware component and a diagnostics signal interpreter. This interpreter evaluates the output from the safety platform and removes the torque if that safety function is activated.

On the assumption that human being lives could depend on the removal of the torque, this stop function must be monitored and described following the industrial standard IEC 61508. This certification standard describes the necessity of adding redundancy features to the exchanged data which diagnose the removal of the torque. This redundancy feature is a safety channel architecture 1oo2, which is shown in Fig. 7.

In spite of the simplicity that this model could involve, it must be stressed that its complexity relies on the safety related add-on designed (shown in Fig. 3). This add-on is developed to Safety Integrity Level

3 (SIL3) according to IEC61508:2010[11], Category 4 and Performance Level e (PLe) according to ISO 13849-1 [17].

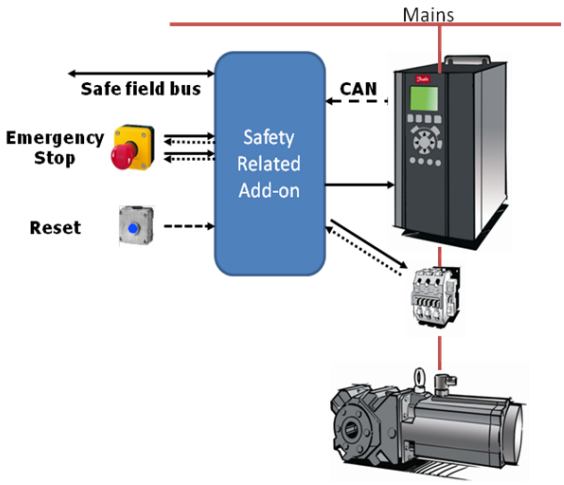

Fig. 3. Danfoss Case Study. The diagram shows the interconnection of different elements of this case study. The case study is developed according to IEC 61508.

This section describes the implementation of this case study using SystemC.

Our SystemC model mixes two different types of modelling: TLM for the highest level of abstraction, in which the communication among the different modules and channels is the most relevant part of the implementation, and the RTL model that simulates the behaviour of the safety platform which includes the utilization of a safety channel module in each core, in order to establish a safe connection between cores.

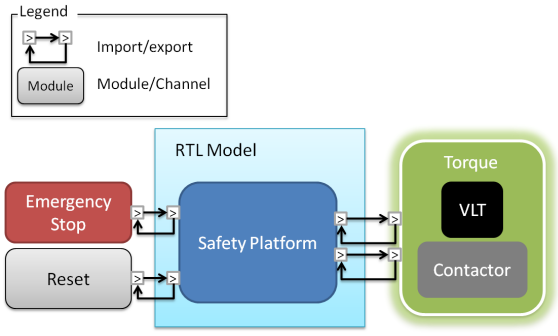

Fig. 4. TLM/RTL Model. Emergency stop, reset, VLT, contactor and torque modules are TLM elements because of the relevance of their communication nature. The safety platform is designed following a RTL model to synthesize the model specifications into FPGAs.

Fig. 4 shows the different components of the model and the two different parts of it. The TLM section involves the Activators, the Controller Area Network (CAN) bus system and the Torque inside the green square. The blue square contains the RTL part of the system, the safety platform, which includes the safety communication between cores.

In the subsections below both TLM and RTL models are explained in detail.

A. TLM Model

The TLM illustrates the message passing behaviour between the different elements of the case study and how they interact at the highest level of abstraction. The main difference between TLM and RTL models is that in TLM modelling there is no need to define the full behaviour of the element at a synthesizable level, it can only be defined as an interchange of messages through ports/channels from/to the interfaces of the components.

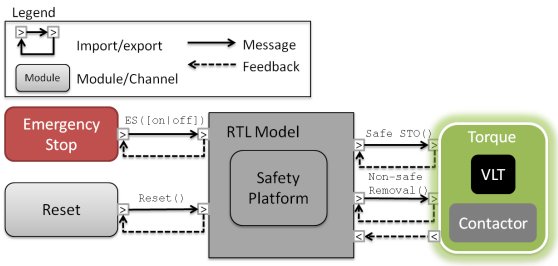

Fig. 5. TLM model. The main objective of TLM models is to study and represent the communication between modules. The use of TLM requires the message passing between all the elements that represent the system.

As it can be seen in Fig. 5, the idea of using TLM for all the elements except for the safety platform is to send messages from the elements that mean interaction with the environment, such an emergency stop button and a reset one. In addition, TLM includes the system that is in charge of getting and evaluating the diagnostic results from the safety platform and means interaction with the Torque element, since in this model, we will only consider this elements as output components, just to give a full understanding of the use of a safety platform.

On the other hand, the decision of using a RTL model for the safety platform emerged from the necessity of describing at the highest level of detail this diagnostic methodology developed by Danfoss, and the possibility to synthesize and generate code directly to a FPGA to verify the specifications of the case study. Moreover, the validation process that involves certification standards requires of good time accuracy results, which are not available when using TLM models. TLM models are mainly used for message payload passing schemas.

B. RTL Model

This RTL SystemC model has been developed on the basis of a Simulink model provided by Danfoss. Due to the dataflow nature of the Simulink version of this case study, we have decided to implement a RTL version of the problem in SystemC, which is closer to signals flow, instead of a TLM model.

At the highest level of abstraction, Danfoss Case Study is described as two different safe cores that handle redundant output signals, one from each processor (Fig. 6). Multicore devices are getting more and more familiar for embedded systems and, with the proper isolation mechanisms for sharing re-

sources, they are a very good architecture for mixed criticality system development. Moreover, each processor receives as inputs to be re-processed the output signals from the other processor through a safe channel 1oo2[11] module, providing this case study with the required redundancy and feedback needed on industry safety systems, as described in the international standard IEC 61508[11].

The next figure describes how the two safe cores of the safety platform interchange data in order to obtain a redundant output that is later used by the system that interacts with the Torque. On the left side it is shown the diagram of the interconnection between the cores as it is described in IEC 61508 and, on the right side, the SystemC model of this interconnection.

These redundant outputs contain the information obtained from the diagnostic stage from each core, joined with the diagnostic output from the other core, and are processed in order to activate the Safety Function, which initiates the Safe Torque Off (STO) signal in case of failure or emergency stop.

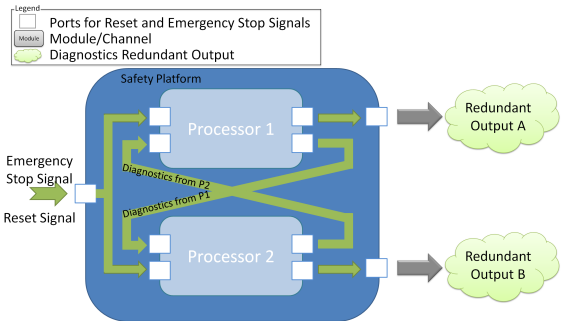

Fig. 6. Safety platform model to achieve redundancy. It represents the interconnection of two cores using a safe channel architecture 1oo2 described in IEC 61508 whose main feature is the redundancy of data.

More precisely, each processor goes through a power up stage that checks the availability and correctness of the system components and through a diagnostic checking stage. The results of this checking stage are forwarded to the other processor in order to verify whether the redundant output from both processors should be enabled or not, which activates the Safety Function.

Both redundant outputs signals that activate or deactivate the Safety Function are connected to the TLM model of a CAN bus system, which is in charge of removing the Torque when the Safety Function is activated and also of providing feedback to the safety platform about the state of the Torque.

Due to the complexity and the relevance of the modules that are modelled within the safety platform, the functionality of each core is described in detail in the next section to provide a proper understanding of the whole document.

B.1 Single Processor Model

Each processor is composed of four modules, which are all interconnected among themselves and that share the input signals related to the emergency stop and the reset. These modules can be easily grouped into three different types of modules according to their functions: diagnostic modules, reset modules and communication modules.

Regarding diagnostic modules, this safety platform includes two types. The first of them is the Power Up Self Test module. This module starts with a complete checking during the power-on of the system and waits for a hardware reset of the system. In addition, it remains activated waiting for the moment the Safety Function is activated, to contribute to a full reset of the system. Apart from this Power Up Self Test module, another diagnostic element resides inside the safety platform: the Diagnostic module. This is the main element among the diagnostic process. It processes signals after the power-on self test stage during the whole run-time of the platform, diagnosing the state of the platform and transmitting it to the other core through the most important element in this platform, the communication channel.

Other modules that are implemented in this safety platform belong to the communication process between cores. Due to the safety-critical nature of the platform, a design methodology must be followed for its implementation following the international standard IEC 61508. This standard describes a safe channel architecture that consists of two safe channels connected in parallel so that each channel can process the safety function, and the applied diagnosis. This module is called Safe Channel 1oo2 (Fig. 7) and is crucial for the diagnostic module because it receives the output from the other processor though the safe channel and it is processed depending on the emergency stop and/or the reset buttons.

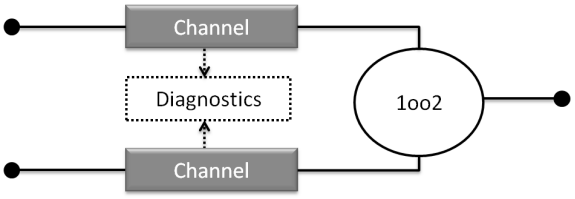

Fig. 7. Safe Channel 1oo2 (Extracted from [11]). Concept of a safe channel 1oo2 architecture. The diagnostic procedure includes the verification of data shared by two different channels leading into a single diagnostic result.

The last module that the safety platform includes is the reset module. This module launches the reset function depending on the state in which the program is and activates the reset function on the system when required. This module is actually an emulation of a status machine inside the SystemC model. Fig. 8 shows the RTL model of a single core of the safety platform.

This developed methodology, joined with the elements designed for the implementation of a safety platform, describes the Danfoss case study. Results will show how this platform, which is modelled using SystemC, achieves the necessary features required by safety-critical platforms as certification standards de-

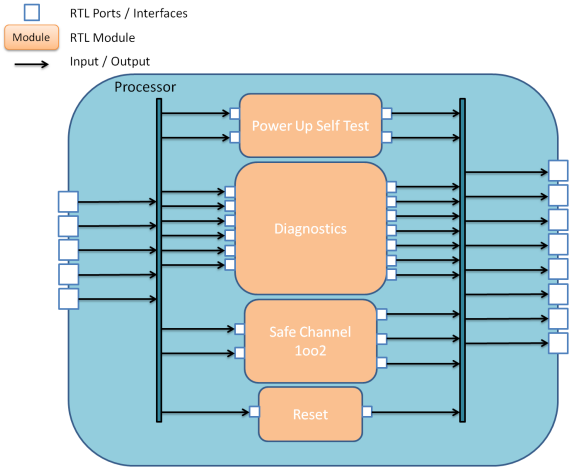

Fig. 8. RTL Model of a Single Core. A single core is made up of four modules: a power up module which starts a diagnostic stage at start up, a diagnostic module that leads the diagnostic process during the run-time, a safe channel 1oo2 that interconnects with the other core and a reset module which controls hardware resets.

scribe.

IV. Experimental Results

The critical feature that involves the development using SystemC, here described within a safety platform, is the possibility of adding redundancy to the diagnostic process that affects the removal of an industrial torque. The two cores go through a diagnostic state in which local variables, joined with external diagnostic processes received through the safe channel 1oo2, are evaluated in order to activate or deactivate the safety function of the torque.

Our approach allows safe communication between two different cores using SystemC modules. As a conclusive proof, Fig. 9 shows the behaviour of the redundant outputs from both core 1 and core 2 that are activated at the same time, which means that there is no delay between core communications. This core safe communication issue is a requirement described in the standard IEC 61508 for the industry domain.

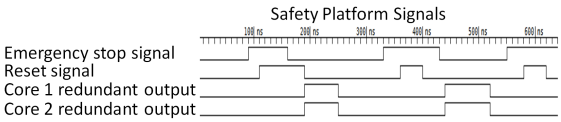

Fig. 9. Core 1 and Core 2 Redundant Outputs. Redundant outputs from both core 1 and core 2 that are activated at the same time, which means that there is no delay between core communications. The fact that both redundant signals are activated will lead into the activation of the safety torque-off function.

On the other hand, the necessity of creating these safety channels 1oo2 in our design results in the possibility of diagnosing whether the activation of the safety function STO, which removes the torque, is required or not. This feature is shown in Fig. 10. Both core 1 and core 2 local STO status signals are local values that are the result of diagnosing whether a STO function is required or not. For this reason, core 1 and core 2 external STO signals describe the values that each core sends out through the safe channel 1oo2 to the other core. Once these external and local values are diagnosed in the diagnostic module, if both external and local values are the same for activating the STO function, the redundant outputs 1 and 2 are generated and sent out to the system in charge of removing the industrial torque.

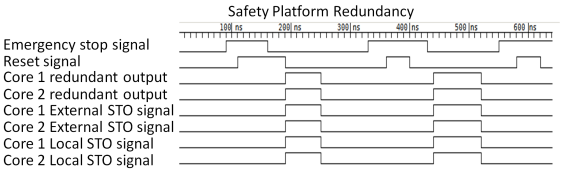

Fig. 10. Core 1 and Core 2 Redundant ouputs and STO signals. Redundant outputs lead into the activation of the safe torque-off function, which is transmitted to the other core through safe channels. Using this safe channel 1oo2 contributes to lack of delay in signals from one core to another.

This approach demonstrates that the utilization of a safe channel to interconnect two cores using SystemC modules, guarantees a good performance between the communications of both cores, since signals do not experiment any delay. Since diagnostic stage in safety-critical systems is crucial, response time should be high. This model ensures that the delay between the intercommunication of two cores through safe channels, following the description detailed in the certification standard IEC 61508, is null.

Due to the behaviour of safe channels and their time response, it is important to notice that SystemC allows the addition to the model of time descriptions which are useful to diagnose processors' features. It is possible to describe the length of an operation within a processor in nanoseconds or even at cycle level, making possible a deep description of the time execution of different tasks in a processor. This makes possible to describe the worst-case execution time and also the best-case execution time, which are very important to evaluate systems.

Extending this concept to safety-critical systems makes possible to study the repercussion and impact of critical and non-critical tasks that could be executed at the same time in processors that share memory. The fact of adding time-execution descriptions to SystemC models brings the possibility of extracting and determining processor features.

V. Conclusions

Our implementation of a safety platform from a real case study using SystemC libraries and different types of development methodologies such as TLM and RTL modelling, in addition to a co-design methodology, provides developers with a set of tools that, despite the fact that SystemC does not include any safety-critical and certification related API, are able to describe, at both high and low levels of abstraction, complex safety-critical systems. Safety-

critical systems are complex systems that need a new methodology that makes possible the utilization of accurate descriptions. Furthermore, a progressive refinement degree of the implementation is required in order to validate the requirements that are necessary for each component within critical systems, such as isolation and time partitioning. For this reason, our RTL model of a safety platform extends critical features to SystemC that does not include by itself, giving rise to the possibility of creating a safety-critical library for further system development.

It seems undeniable that SystemC is a powerful language that can be used to describe, develop and verify the main features that the validation progress of safety-critical systems development involves. Furthermore, co-design brings the opportunity to design hardware and software separately, despite requirements come from a common description, such as certification standards.

Although the RTL modelling of a safety platform assumes a complete development process and can be completely verified, there are some enhancements for its development.

One of these enhancements is the improvement of the TLM model, including the RTL model we have earlier described and developed into a working TLM model in order to verify the message passing through channels and ports before synthesizing the system into a FPGA. Since the TLM model that we have developed at this moment is still being improved, the final implementation of TLM and RTL models working together is one of our ongoing objectives. Furthermore, synthesizing this model into digital hardware would be the last step in the development process following a co-design methodology to verify a proper behaviour of the platform.

Another future work is the addition of a run-time monitoring module within the system. Run-time monitoring elements avoid the violation of a fixed set of conditions (accessing to protected memory regions by non critical modules, over-intensive use of shared devices, out of range parameters, etc.) that have been previously defined between elements that are using the same communication channel (bus, signal, etc.). More specifically, this new module can be added to the hardware design as a new library element in both RTL and TLM models, due to the advantages of monitoring the outputs and inputs of modules. By adding this new feature, the torque that dispatches the information about its status through the CAN bus directly to the safety platform, would be intercepted by the run-time monitoring avoiding the violation of address memory or other modules inside the safety platform. In this manner, we would have developed a new library that provides SystemC designs with a model of a safety platform which includes a safety redundant channel 1oo2 architecture described in IEC61508, joined to run-time monitoring components that avoid memory access violation.

Acknowledgments

This work has been supported by the Artemis JU project RECOMP: Reduced Certification Costs Using Trusted Multi-core Platforms (Grant Agreement number 100202).

References

[1] The Open SystemC Initiative , *Retrieved May 10th, 2012, from http://www.accellera.org/downloads/standards/systemc/*
[2] J. Bhasker, *A SystemC Primer.* Allentown PA : Star Galaxy Publishing, 2002. ISBN 0965039188.
[3] David C. Black ... [et al.], *SystemC: From the Ground Up.* New York : Springer, 2010 ISBN 9780387699585.
[4] Xilinx Embedded Development Kit, *Retrieved May 10th, 2012, from http://www.xilinx.com/tools/platform.htm*
[5] Qsys - Altera's System Integration Tool, *Retrieved May 10th, 2012, from http://www.altera.com/products/software/quartus-ii//qts-qsys.html*
[6] Xilinx Zynq - Extensible Processing Platform, *Retrieved May 10th, 2012, from http://www.xilinx.com/products/silicon-devices/epp/zynq-7000/index.htm*
[7] Altera SoC FPGA , *Retrieved May 10th, 2012, from http://www.altera.com//soc-fpga/proc-soc-fpga.html*
[8] Catapult C Synthesize Tool, *Retrieved May 10th, 2012, from http://www.mentor.com/esl/catapult/*
[9] Vivado Design Suite, *Retrieved May 10th, 2012, from http://www.xilinx.com//vivado/*
[10] G. De Michell and R. Gupta, *Hardware/Software Co-Design.* Proceedings of the IEEE, vol. 85, 1997.
[11] Functional safety of electrical/electronic/programmable safety related systems. *Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3. BSI Standards Publication. BS EN 61508-6:2010.*
[12] DO-178B, *Software Considerations in Airborne Systems and Equipment Certification.*
[13] ISO 26262, *Road vehicles - Functional safety.*
[14] T. Grötker, S.Liao, G. Marin, and S. Swan, *System Design With SystemC.* Kluwer Academic Publishers, 2002.
[15] Danfoss Power Electronics, *Retrieved May 10th, 2012, from http://www.danfoss.com/*
[16] RECOMP Project, *Retrieved May 10th, 2012, from http://www.recomp-project.eu/*
[17] ISO 13849-1, *Safety of machinery.*