Dipl.-Ing. Gerhard Grießnig

FSAR
A Fail-Safe Architecture for Reconfigurable Programmable Logic Devices

---

Dissertation

vorgelegt an der
Technischen Universität Graz

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften
(Dr. techn.)

durchgeführt am Institut für Technische Informatik
Technische Universität Graz
Vorstand: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Reinhold Weiß

Graz, im Mai 2011

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Kurzfassung

Die zunehmende Komplexität von eingebetteten Systemen und der steigende Kostendruck stellen vor allem die Hersteller von sicherheitskritischen Systemen vor immer neue Herausforderungen. In dieser Dissertation wird eine neuartige fehlersichere Architektur für rekonfigurierbare, programmierbare digitale Bauelemente (FSAR) vorgestellt. Im Gegensatz zu derzeitigen sich auf dem Markt befindlichen Lösungen, die hauptsächlich Mikrokontroller basierend sind, ist die hier vorgestellte Lösung ausschließlich auf programmierbarer Hardware basiert.

Ein weiterer Schwerpunkt neben der Architektur liegt in der Methodologie, die zum Nachweis der funktionalen Sicherheit und der geforderten Sicherheitsintegrität erforderlich ist. Im Kontext dieser Arbeit wurde ein Verfahren zur Durchführung von Fehler-Injektionstests entwickelt. Die fehlersichere Architektur und die Methodik wurden von einem unabhängigen Zertifizierungsinstitut begutachtet und für einen Einsatz gemäß SIL 3 nach IEC 61508 bzw. IEC 61800-5-2 sowie Kategorie 4 Performance Level e nach ISO 13849 als tauglich befunden. Ein entsprechender Prototyp für einen industriellen Einsatz in der Automatisierungstechnik wurde entwickelt und evaluiert.

Abstract

The increasing complexity of embedded systems and the growing cost pressure are imposing new challenges for the producer of safety-critical systems. Within this thesis, a novel **Fail-Safe Architecture for Reconfigurable Programmable Logic Devices** (FSAR) is described. In contradiction to commonly used solutions on the marked which are mainly realized using microcontrollers, the presented solution is based on reconfigurable programmable hardware only.

Additionally to the architecture, a further focus is set on the methodology to ensure the fulfillment of the functional safety and the required safety integrity requirements. In this context, a method to perform fault insertion tests on the target architecture was developed. The fail-safe architecture as well as the methodology were assessed by an independent assessment body and is applicable for the usage of SIL 3 applications according to IEC 61508 respectively IEC 61800-5-2 as well as category 4 and performance level e according to ISO 13849. A corresponding prototype for industrial usage in the automation domain was developed and evaluated.

Danksagung

Für meine Doktorarbeit schulde ich sehr vielen Menschen einen herzlichen Dank. Ein besonderer Dank gilt meinem Doktorvater Herrn O. Univ. -Prof. Dipl. -Ing. Dr. techn. Reinhold Weiß für die akademische Führung und die abwechslungsreichen Einblicke in die Wissenschaft von einst und heute. Des Weiteren möchte ich mich speziell bei meinem Projektleiter Herrn Ass. -Prof. Dipl. -Ing. Dr. techn. Christian Steger für seine außergewöhnliche Unterstützung in organisatorischen und wissenschaftlichen Fragen bedanken.

Meine Dissertation wäre nicht möglich gewesen ohne die Unterstützung der SIEMENS AG und AVL GmbH. Vor allem danken möchte ich Dr. Herbert Tanner und DI Joachim Wahrbichler, die mir den Start dieser Arbeit ermöglicht haben, aber auch Dr. Theodor Sams sowie Dr. Josef Affenzeller, die mich für einen erfolgreichen Abschluss unterstützt haben.

Einen besonderen Dank möchte ich meinem Diplomanden und späteren Dissertanten DI Roland Mader aussprechen, der mich über all die Jahre an diesem Thema begleitet und tatkräftig unterstützt hat.

Mein tiefster Dank gilt jedoch meiner lieben Frau Doris, die mir die Zeit, das notwendige Verständnis und in schwierigen Phasen auch die erforderliche Motivation gegeben hat, um diese Arbeit beenden zu können. Auch meinen Sohn Manuel und meine Tochter Elina möchte ich hier nicht unerwähnt lassen. Sie haben auf so manche gemeinsame Stunde verzichten müssen.

Graz, Mai 2011

Extended Summary

The increasing complexity of embedded systems and the growing cost pressure are imposing new challenges for the producer of safety-critical systems. Therefore, the industry is continuously obliged to search for improved methods and new technologies. In this spirit, this work was motivated by the fact that industry demands cost-efficient concepts for the realization of safety functions for industrial automation.

In particular, for the SIEMENS product family SIMOREG© DC-Masters [AG10], which are applicable for single- and four-quadrant DC drives, a redevelopment is necessary due to discontinuation of hardware components. In the course of this redevelopment, the SIMOREG© DC-Master product family shall be extended by the safety-critical stop functions *Safe Torque Off* and *Safe Stop 1* as defined in the functional safety application standard **IEC 61800-5-2** for *Adjustable Speed Electrical Power Drive Systems*. A conducted market analysis shows that most applications of a SIMOREG© DC-Master require a safety integrity level (SIL) of at least 2 according to the IEC 61800-5-2.

Taking into account the top level requirements mentioned above, the goal of this thesis is an elaboration of an entire *safety concept* for a **failsafe system**. The assessment of this concept by an independent certification body was aspired. Additionally to the requirements of the imposed safety standards, the safety concept shall consider the economic facts of the product development. Economic facts are considered in this thesis as (1) time to market, (2) product and development costs as well as (3) the experience and know how of the department where the system shall be developed, produced and maintained. Special attention should be paid to (1) due to the fact that the desired solution is in competition with external measures to realize safe stops in today's automation systems with DC motors and (3).

Within this thesis, a **Fail-Safe Architecture for Reconfigurable Programmable Logic Devices** (FSAR) is elaborated. In contradiction to commonly used solutions on the marked which are mainly realized via micro-controllers, the presented solution is based on reconfigurable programmable hardware only. The presented novel FSAR was patented by SIEMENS [GFH+09] in 2008 and was published as a full paper selected for the IEEE conferences. The fail-safe system contains two channels (HFT=1). Each channel is able to perform the safety functions independently. Thus, if a channel is affected by a fault, the other channel is still able to execute the safety functions. The entire system enters the safe state if a faulty channel is detected. Figure 1 illustrates the fail-safe system. It comprises the following entities:

- The terminals $\mathbf{SF1}_1$, $\mathbf{SFm}_1$, $\mathbf{SF1}_2$, $\mathbf{SFm}_2$ are connected with an external device (e.g. a control panel) that can be used to activate the safety functions.

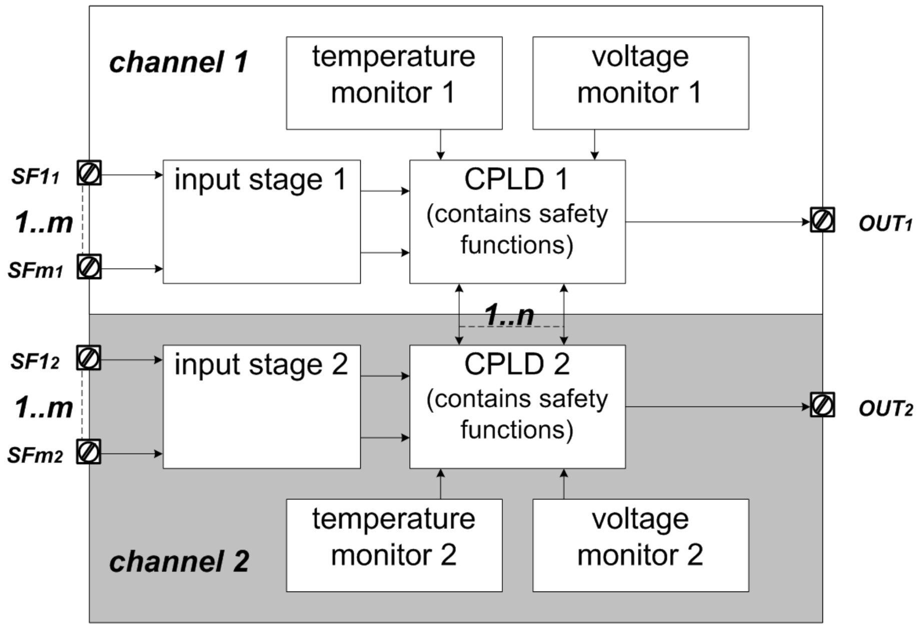


Figure 1: Fail-safe Architecture for Reconfigurable Programmable Logic Devices

- The **input stages** represent an interface for the activation of the safety functions that assures electrical isolation by optocouplers.
- The two **CPLDs** realize the safety functions. In order to be able to perform diagnostic checks, the CPLDs exchange signals. The CPLDs control the safety-critical outputs (OUT_1 and OUT_2) of the fail-safe system. Each CPLD is clocked by a separate external oscillator.
- The terminals $\textbf{OUT}_1$ and $\textbf{OUT}_2$ are connected with an electric motor to allow power to be applied to the motor or to respectively shut down the motor.
- Each channel contains a **temperature monitor** to detect temperature deviations from a specified range.
- There is one **voltage monitor** for each channel which detects whether the supply voltage is leaving a specified range. Furthermore, both channels of the fail-safe system are protected against dangerous overvoltage.

Due to the fact that it is not sufficient to introduce a novel safety architecture without pinpointing the way to ensure that this architecture fulfills the requirements of the imposed safety standards, an additionally focus was put on the **related methods** to satisfy the imposed safety requirements. Therefore, experiences and hints from other domains as well as upcoming standards, regarding the design, implementation and tool chain, were considered to achieve an adequate level of safety and safety integrity for the presented FSAR.

The applicability of this safety concept for programmable logic devices (PLD, CPLD, FPGAs) has been proved in terms of safety and cost-efficiency by its utilization for an industrial prototype (see Figure 2) for an industrial power drive application.

The safety concept and its application were reviewed and assessed by an independent certification authority (TÜV SÜD). TÜV SÜD stated that the presented concept is suited to achieve **SIL 3** in adherence to IEC 61508 and IEC 61800-2 as well as **Cat 4, PL e** in adherence to EN ISO 13849.

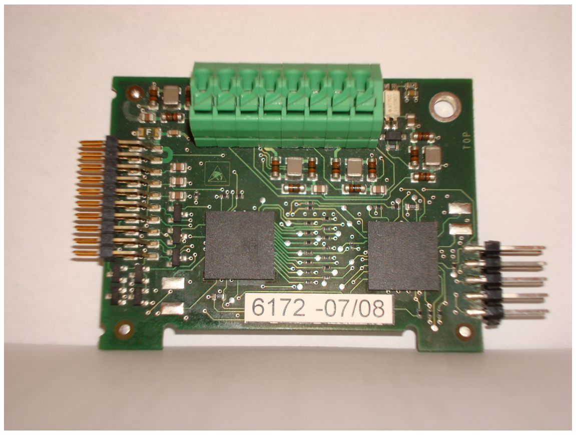


Figure 2: F-Module

Furthermore, this thesis elaborates a method which allows the application of a proven verification method (fault insertion testing) for a PLD-based fail-safe system. Methods to verify comparable micro-controller based systems using fault insertion testing are not applicable for the fail-safe system because it contains exclusively PLDs. It is possible to verify the safety integrity measures of the fail-safe system using the system hardware. The successful verification of the realized safety integrity measures using fault insertion testing is an important prerequisite for the certification of the developed PLD-based fail-safe system in adherence to the imposed standards. A full paper describing this methodology was published and presented at the IEEE International Conference for Design, Automation and Test (DATE Conference 2009).

Finally, it needs to be underlined that the applicability of the presented "Fail-Safe Architecture for Reconfigurable Programmable Logic devices" (FSAR) is **not limited** to the automation domain. It can be concluded that the use of a PLD-based (PLD, CPLD or FPGA) safety concept is a competitive alternative to the use of micro-controller-based safety concepts if comparably uncomplex safety functions need to be realized. In this case, the comparatively simple functionality does not justify a software implementation including the great effort for development of complex software safety integrity measures to make the use of micro-controllers acceptably safe.

Contents

List of Figures

List of Abbreviations

ABS	Anti-lock Braking System
ACC	Adaptive Cruise Control
ASIC	Application-Specific Integrated Circuit
ASIL	Automotive Safety Integrity Level according to ISO 26262
ATTEST	Advancing Traffic Efficiency and Safety through Software Technology
AUTOSAR	AUTomotive Open System ARchitecture
Cat	Category according to ISO 13849-1
CBD	Component Based Development
CCF	Common Cause Failure
CESAR	Cost-efficient Methods and Processes for Safety Relevant Embedded Systems
CMF	Common Mode Failures
COTS	Commercial of the shelf (hardware or software) products
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DATE	Design, Automation and Test in Europe
DC	Diagnostic Coverage
E/E/PE	Electric / Electronic / Programmable Electronic
EAST-ADL	Electronics Architecture and Software Technology - Architecture Description Language
ECU	Electronic Control Unit
EGAS	Electronic Gas Pedal
F-Modul	Fail-safe Modul
FMEA	Failure Mode Effect Analysis
FMEDA	Failure Mode Effect Defect Analysis
ESC	Electronic Stability Control
FPGA	Field-Programmable Gate Array
FSAR	Fail-Safe Architecture for Reconfigurable Programmable Logic Devices
FSM	Functional Safety Management
H&R	Hazard Analysis and Risk Assessment
HFT	Hardware Fault Tolerance
HDL	Hardware Description Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineering
IFIP	International Federation for Information Processing
IP	Intellectual Property
ISO	International Organization for Standardization
PDS	Power Drive System
PIF	Power Interface
PFD	Probability for Failure on Demand
PFH	Probability of a Dangerous Failure per hour
PLD	Programmable Logic Device
PL	Performance Level according to ISO 13849-1

RE	Requirements Engineering
RAM	Random-Access-Memory
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RTP	Reference Technology Platform
SE	System Engineering
SFF	Safe Failure Fraction
SIL	Safety Integrity Level according to IEC 61508, IEC 61800
SoC	System-on-Chip
SRS	Safety-related System
SS1	Safe Stop 1
STO	Safe Torque-Off
SysML	Systems Modeling Language
TMR	Triple Modular Redundancy
UML	Unified Modeling Language
V&V	Verification & Validation
VLSI	Very-Large-Scale Integration
XML	Extensible Markup Language

Chapter 1

Introduction FSAR

1.1 Motivation

Embedded systems technology and applications have been evolving at a fast pace, illustrated for example through the evolution of vehicles that have been transformed into embedded computing systems with hundreds of embedded computing devices and several networks. Today, about 3 billion embedded units are delivered per year, and the world market for embedded systems encompasses approximately 160 billion Euros with an annual growth of about 9 percent [EJ09], [ES09].

In this context, terms like *safety, diagnosability* and *dependability* as well as *reliability* are gaining importance for the development of electronics and electronic programmable systems. This fact is not limited to one single domain, it influences nearly all fields of applications dealing with the development of embedded systems. In particular, all specialist disciplines or manufacturers which are developing components or products for the usage in *safety-critical* systems are forced to deal with these terms.

To better understand what **safety-critical** means in an embedded system we consider the evolution of an ordinary road vehicle as example. Only a few decades ago, embedded systems in vehicles were very rarely applied with almost no software inside. With the electronic torque control (EGAS) [ABD$^+$07] and the further development of microchip technology, more and more functions were realized via embedded systems. Today's vehicles were provide a high number of various visible and invisible functions for the driver to increase the comfort but also the safety of the vehicle passengers.

An airbag which is a well-known vehicle safety device may serve as an example. It is a protection device consisting of a flexible envelope designed to inflate rapidly during a vehicle collision, to prevent occupants from hitting interior objects such as the steering wheel or a window. This means that the proper function of an airbag supports the protection of occupants in a vehicle in case of a crash. However, an unintended activation of the airbag during driving can cause a major accident and has to be avoided.

The airbag is only one function in a today's vehicle which is safety-critical. There are many other functions like anti-lock braking system (ABS) or the electronic stability control (ESC), which have a direct impact on the driveability and behavior of a vehicle. Furthermore, a lot of new technologies like adaptive cruise control (ACC) or vehicle connectivity are entering the automotive market. This continuous electrification significantly increases

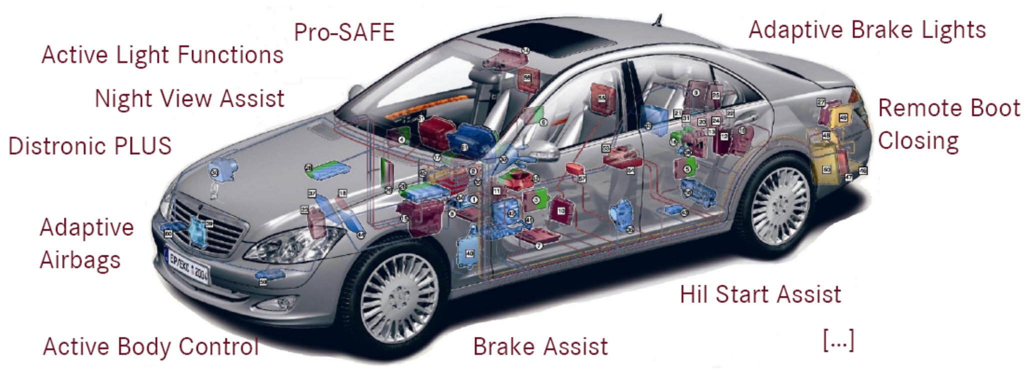


Figure 1.1: The figure shows examples of safety-relevant functions and the complexity with the high numbers of electronic control units (ECU) that control one or more of the electrical systems or subsystems in a modern vehicle. Figure from [Con11] (Daimler).

the number of electric and electronic components such as sensors, actuators and electronic control units (ECU) as well as their corresponding lines of program code. Current road vehicles are running with 100 of million lines of software code [Mos10] that have been developed conjointly by large teams from different organizations.

It is evident that the numerous functions, technologies and the distributed development directly influence the *complexity, quality*, *costs* and *time to market* of a vehicle. But this is by no means the end of the story in the automotive domain. The automotive industry shares the view that in the next 10 years, 90 percent of its expected innovations will be based on electrical/electronic systems with a huge emphasis on the safety-related systems [Mos10].

Nowadays, a lot of research effort is spent on new technologies to reduce emissions and to continuously improve the vehicle safety. Today's catchphrases like *hybridization, electro-mobility* as well as *car to car* or *car to infrastructure communications* are increasing the complexity of future vehicles directly impacting both quality and costs. To ensure that the new vehicles ensure an acceptable and comparable level of safety, new standards like the ISO 26262 [Int10] were developed to set requirements to the development process and the system itself depending on the criticality of the realized safety functions.

Currently existing methods, processes and tools need to be improved, and new technologies are required to manage the challenges of increasing complexity and costs by ensuring an appropriate product quality and taking into consideration the system properties *safety, diagnosability*, *dependability* and *reliability* as mentioned above.

These challenges are commonly shared in nearly all domains working in the embedded systems area. Currently, partners from automotive, aerospace, rail and the automation domain are working together in a large European research project called CESAR [GMP$^+$10] "Cost-efficient methods and processes for safety-relevant embedded systems" to exchange experiences and know-how. The aim of the project is to improve safety-related processes and methods within their domains and to find commonalities in order to increase applicability of methods and tools to save costs.

It is the task of the research community to continuously search for new technologies and improved methods to answer today's problems. With this spirit, this thesis was started to find a "best solution" for a concrete problem definition in the automation domain. Account must be taken of the application of new technologies in safety-critical systems, in particular when such technologies are not considered in the domain specific safety standards. Therefore, such an application requires an investigation of further, domain-external safety standards to ensure the *state of practice* and to achieve an "adequate" level of safety. This approach was considered for this thesis and for the development of a corresponding prototype.

1.1.1 Terminology

In this subsection, the definition of the safety-relevant terms *safety, diagnosability* and *dependability* are discussed. These definitions were aligned with a huge effort between safety experts from the aerospace, automotive, automation and rail domains within the task force safety & diagnosability in the CESAR project [BGA+10]. This task force consists of high-level experts from industrial end users, tool vendors as well as academics who guarantee a broad acceptance and an adequate review of the **publicly available** definitions within the CESAR deliverable. The following definitions of *safety, diagnosability* and *dependability*, defined in [BGA+10], will be reused and are valid for the entire thesis.

Safety

Safety is defined as the ability of a *system* to not cause or contribute to the occurrence or aggravation of *harm*.

- **System** is taken here in its most wide and generic meaning, for instance as defined in [ALRL04], an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena. These other systems are the environment of the given system. The system boundary is the common frontier between the system and its environment.
- **Harm** is defined as death, physical injury or damage to the health of people (up to, at least for some application domains and standards, damage to property or the environment).

In this context, the overall goal associated with safety is to transform the severity and likelihood of a risk inherent in all human activity to lower, acceptable levels. In particular, it deals with system safety, i.e., the application of special technical and managerial skills and procedures in a systematic, forward-looking manner in order to identify and control hazards throughout the entire life cycle of a project, program or activity. A generic system can be defined as a group of interrelated processes and functions composed of people, procedures, equipment, material, tools, facilities and software, operating in a specific environment to perform a specific task or mission. In many domains, safety is managed according to regulations enforced by national or international authorities. In most of the regulated domains such as civil aviation, this is achieved through a certification or qualification process. *Certification* is defined as "the legal recognition that a product, service, organization or person complies with the applicable requirements". It is also defined (EN 45020:1993) as "the procedure by which a third-party gives written assurance that a product, process or service conforms to specified requirements".

An important item related to safety is the notion of *criticality*. This notion as used in the context of safety should not be confused with the criticality of a risk, measuring a combination of its likelihood of occurrence and the severity of its consequences. In this case, criticality is not associated with a risk but an *item* which can be interpreted as the system or any actual or abstract part of it like function, sub-system, equipment, component, hardware or software – any entity that may fail. The criticality of an item is a measure of the severity of the consequences of the potential failures of the considered item (alone or possibly in combination with other events, e.g. exposure). It is defined without consideration of the likelihood of occurrence of the potential failures of the item (or of the other considered events).

The criticality (it may be called, according to the standards applicable to the various domains, class, development assurance level, safety integrity level) is the basis for the safety framework that may be summarized as follows:

- Top level feared events are identified, analyzed and classified according to the severity of their consequences and to a predefined ordered scale of categories of effects;
- Criticality categories are allocated to the system and to its functions and hardware and software elements, according to the most severe category of the feared events that the potential failure of the considered item could cause or contribute to (the "degree of causality" may be taken into account in the allocation of criticality category);
- A set of development and validation rules is associated with each criticality category:
  - Some of these rules are generic and concern the processes, methods and techniques: They aim at adapting the development and validation effort and level of rigour in accordance to the criticality category;
  - The rules put bounds on the minimum number of barriers against fault propagation or on the maximum probability of occurrence of the concerned top level feared event (generally restricted to the occurrence due to some physical random fault, explicitly excluding design, software or human operations).

Diagnosability

Diagnosability is defined as the ability of a system to support the identification of its status with respect to its potential *faults*.
The above intentionally definition is left as open as possible. In practice, one has to define more accurately in particular what kind of information is to be identified. This depends on the objectives and context of the diagnosis. In a pragmatic way, what is needed is not

"any kind of information" about the fault(s) but the necessary and sufficient set so as to unambiguously identify the best actions according to the objectives and current context. In other words, it is considered that the diagnosis is less interested in the identification of the problem, than (1) of the existence (or not) of a problem and (2) of the solution. For instance, if an equipment fails, a first-level diagnosis is limited to identify which one, so as to know which one to replace. Some second- (third-, etc.) level diagnosis will then be useful to know whether the equipment has actually and permanently failed, which component has failed (e.g., to be replaced so as to repair the equipment) and even more detailed information about the fault in terms of localization, origin, propagation, effects, occurrence rate, to be able to e.g., devise adapted procedures, rules, design, and assessment for other existing and future systems.

In the following paragraphs it is assumed that there is a clear distinction between the identification of the actions (to solve the problem) and their realization. In practice, this is not always the case and there are situations in which the diagnosis is performed through the execution of some recovery or repair actions. In other words, the recovery or repair actions are not first identified (diagnosis) and then realized, but some are attempted, and it may be only when the problem is solved that the diagnosis is completed. The few basic notions about diagnosability are introduced below in the situation of "separated diagnosis" for clarity, but their transposition, if needed, to the "mixed diagnosis and recovery" situation is straightforward. Please note that the question of who performs which part of detection and diagnosis (the system, an operator, a repairman) is not explicitly addressed since it is focused on diagnosability understood as the capability of the system to provide support to the diagnosis, covering in principle all possible cases of allocation.

The notion of diagnosis can differ between *corrective* and *preventive* diagnosis:

- **Corrective** diagnosis is performed after the explicit occurrence and detection of an anomaly in the system. In this case, the anomaly is the starting point of the diagnosis that consists of distinguishing it from all other possible anomalies for which a different resolution procedure is appropriate.

- **Preventive** diagnosis is performed according to a defined plan. The objectives are to investigate the system state and capabilities as thoroughly as possible so as to detect potential dormant faults or latent errors that could prevent the system from actually meeting its safety and dependability requirements. In general, the preventive diagnosis procedure and plan are defined as part of the justification that the dependability and safety requirements are met and continuously preserved along the operational phase of the system's life cycle.

Note: There is an intermediate category when the diagnosis of a system is initiated not after the detection of an anomaly of that system but after one or several anomalies of other systems or some stressful out-of-specifications conditions applied to the system (e.g., shock, vibrations, lightning etc.).

Independently from the distinction between corrective and preventive diagnosis, one may also distinguish between *in-workshop* and *in-situ* diagnosis:

- **In-workshop** diagnosis: The system is dismounted, removed from its operating environment and brought to a specific place with dedicated tools. In principle, the diagnosis benefits from better capabilities to investigate the system state and condition. However, some anomalies related to the interactions between the system and its environment may be difficult to reproduce and analyze.

- **In-situ** diagnosis: The system is kept in its operational environment, either *on-line* or *off-line*:
  - **Off-line**: The service provided by the system is stopped, either as a direct and automatic consequence of the anomaly or as a subsequent explicit decision and action so as to limit the severity of potential consequences or to facilitate the diagnosis.
  - **On-line**: The system is maintained active with at least some minimal service provided to its users. In general, this decreases the constraints on the time allocated to diagnosis but strongly limits the investigation capabilities because the observable information is mostly imposed by the system activity rather than controlled so as to facilitate the diagnosis.

It is worth noting that the notion of diagnosability is closely related, but not identical, to the classical notions of *observability or monitoring* and *commandability* in automatic control. In some sense, diagnosability may be seen as an extension of the notion of observability when considering the presence of potential faults. The relationship to commmandability of the system (in presence of faults) is twofold. On the one hand, the commandability determines the set of actions that can be performed to get additional information about the fault(s) in the system, and on the other hand, the commandability determines the set of actions that can be performed so as to solve the anomaly.

The latter point is important because as mentioned at the beginning of this section, a diagnosis procedure is not intended to determine all possible information about an anomaly but only what is sufficient to know what to do, i.e., to distinguish it from any other possible anomaly for which the resolution procedure is different. It even appears some how paradoxically that the less commandable a system is (in presence of faults), the easier the diagnosis is since almost all anomalies will be processed by one among a very reduced set of procedures (or even a single one, e.g., stop the train, switch the satellite to survival mode etc.). In a quite restricted meaning, one could limit diagnosability to a particular observability property defined as the capability to distinguish any two different cases of faults in the specified fault model (or at least, any two ones for which the subsequent recovery or repair processing is different, as explained previously). However, this is likely to be insufficient to be usable in practice because the needed observable information depends on the diagnosis procedure. In a similar way, it is important to extend the scope to the objectives of the diagnosis (because they impact the diagnosis itself) and even to the implementation of the recovery actions (because they may be partly interleaved with the diagnosis and because the way they are implemented may have an impact on the requirements and criticality of the diagnosis).

Dependability

Dependability is largely (though not unanimously, see below) understood as an integrating concept and a scientific discipline (the so-called "science of failures") encompassing the whole set of (computer-based) system properties and means considering threats i.e., possible adverse events and situations. This has emerged in the 80's pushed forward by working groups within the IEEE and IFIP [ALRL04]. In this context dependability is defined as "the ability of a system to deliver a service that can be justifiably trusted". The threats to dependability are modeled as a recursive chain comprising:

- **Failure:** An event that occurs when the delivered service deviates from correct service (not necessarily "specified service"; the reference to "correct service" includes the case where the service was not correctly specified); Failures may be characterized by various attributes:
  - **failure modes:** Silence or stop failure, inconsistent (so-called "Byzantine") failure etc.
  - **severity:** Catastrophic, critical etc.
  - **characterization:** Of the failure occurrence process etc.
- **Error:** The part of the total state of the system that may lead to its subsequent failure; an error may be latent or detected
- **Fault:** The adjudged or hypothesized cause of an error; this particularly includes the case of the failure of another system interacting with the considered one, or the failure of a component, hence the recursive nature of this chain. A fault may be in an active or dormant state (i.e., it cycles between states in which it produces, or not, errors). A set of means can be used including:
  - **Fault prevention** means to prevent the occurrence or the introduction of faults
  - **Fault tolerance** means to avoid failures in the presence of faults
  - **Fault removal** means to reduce the number and severity of faults
  - **Fault forecasting** means to estimate the present number, the future incidence and the likely consequences of faults.

These means must be combined and used appropriately with respect to both the threats and to the dependability properties that are relevant for the considered system. In the general dependability framework, dependability comprehensively encompasses all properties such as reliability (continuity of correct service), availability (readiness for correct service), safety (absence of catastrophic consequences on the user(s) and the environment), integrity (absence of improper system alterations) and maintainability (ability to undergo modifications and repairs) [ALRL04].

In this context, **fail-safe systems** are defined as systems that are required to fail in a way to not harm people, the environment or property in case of a failure. Furthermore, fail-safe systems have the well-defined task to achieve and maintain a safe state in case of a fault.

This vision of dependability is now largely shared and used in numerous domains, even if some differences may be noticed, for instance about the utilization of such or such a word (e.g., operator fault, software failure, etc.). In general, these are minor points, not justifying renouncing to this sound, consistent and comprehensive framework. It is worth noting, however, that there still is a more important difficulty about whether safety may be considered as a part of dependability or not.

1.1.2 An Industrial Application

In order to develop a successful product for an industrial or commercial application, a lot of facts, risks and boundary conditions have to be taken into account. One good practicely, which is considered in this thesis is the application of the system engineering process [Bla08], [Wei05]. The system engineering process describes the entire product life cycle from the problem definition to the system retirement and material disposal. This approach includes the initial definition of the problem (to be solved), the identification of a consumer need, the execution of a feasibility analysis, the development of system-operational requirements, the maintenance and support concepts, the accomplishment of a functional analysis allocation of requirements and the development of the top-level architecture for a given system. In the initial requirements definition phase, the target market and the field of application have to be taken into account. In particular, the field of application may require special environment conditions, or the target market has dedicated regulations which are expressed in laws or standards. These requirements can directly influence the product properties and need to be considered in feasibility studies or prototypes.

The underlying motivation of this thesis is to investigate existing safety architectures and to identify or to invent a suitable safety architecture which complies product requirements and considers the boundary conditions of the customer and his experience in processes and technologies. The basic product requirements, the target market and the field of application are predefined. The product shall be applicable for a world-wide usage in the automation sector in which the field of application are *power drive systems*.

For this project, the customer is an in-house department in the SIEMENS AG that mainly develops the hardware for electronic systems. The process for the software development is completed and matured for the department during the SIMOREG-plus Project (see Chapter 1.2.2).

1.2 FSAR

As mentioned above, a lot of different ways are possible to solve a defined problem. For the development of a safety-critical system it is not enough to provide a solution only, decisive is the way of how to develop such a system. This means in particular, which safety standards and corresponding processes will be applied, which methods and tools will be used or which analyses and tests will be performed at which point in time. Usually, functional safety processes require independent audits at dedicated points in time to verify that all activities and methods have been applied at a certain development stage. The purpose is to avoid the occurrence of systematic failures in the development. This is, by the way, the reason why a certification process has to start at the beginning of the product development.

Due to this fact, it is not sufficient to introduce a novel safety architecture without pinpointing the way how to ensure that this architecture fulfills the requirements of the desired safety standards. Therefore, the major outcomes of this thesis are both the developed **Fail-Safe Architecture for Reconfigurable Programmable Logic Devices** (FSAR) as well as the **related methods** to satisfy the safety requirements.

1.2.1 CESAR Project

The CESAR project [Con09,GMP+10] deals with **C**ost **e**fficient methods and processes for **sa**fety-**r**elevant embedded systems. CESAR is a European project funded by Artemis Joint Undertaken and national authorities and regroups 55 partners coming from industry, tool vendors and academics. The global budget amount to approximately 58 million Euros and a cumulated effort of approximately 427 man-years for a duration of three years. CESAR's main objective is the reduction of costs for the development of safety-critical systems which shall be achieved by the improvement of the corresponding processes and methods for the development of safety-critical embedded systems. The approach relies on the establishment of a Reference Technology Platform (RTP) [AGZ+11], providing a conglomerate of modules which facilitate the flexible creation of integrated development environments for the development of safety-relevant real-time embedded systems for various domains. Furthermore, CESAR will bring significant innovations to the two following system-engineering disciplines: (1) requirements engineering (RE), in particular by formalizing multi-viewpoint, multi-criteria, and multi-level requirements, and (2) component-based development (CBD) applied to design space exploration comprising multi-view, multi-criteria and multi-level architecture trade-offs. These innovations are driven – and will be evaluated – by pilot applications from the domains of automotive, aerospace, rail and automation respectively.

The main focus of the CESAR project is set on the left top side of a V-Model (see figure 1.2) in the system-engineering discipline. A further potential can be found in the CESAR *multi-discipline* and *multi-domain* approach. In the *multi-discipline* approach, CESAR requires a close collaboration between the requirements engineering, component based development and the establishment of a seamless tool chain (RTP) by taking into account safety & diagosability and product-line engineering. With the *multi-domain* approach, the aerospace, automotive, automation and railway domains are jointly working together to share experiences and know-how in the development of safety-critical systems in order to identify common methods and tools to finally reduce the development costs.

With the size and quality of the consortium, one expected outcome are orientations for the system engineering of the next generation and the first step towards an open interoperability standard for tool connections.

The CESAR project contributes to the development of FSAR with two major aspects. First, CESAR works as high-level platform to discuss and exchange experiences in the application of "new" technologies and second, the CESAR project supports the argumentation of the safety methods and seamless applied tool chains.

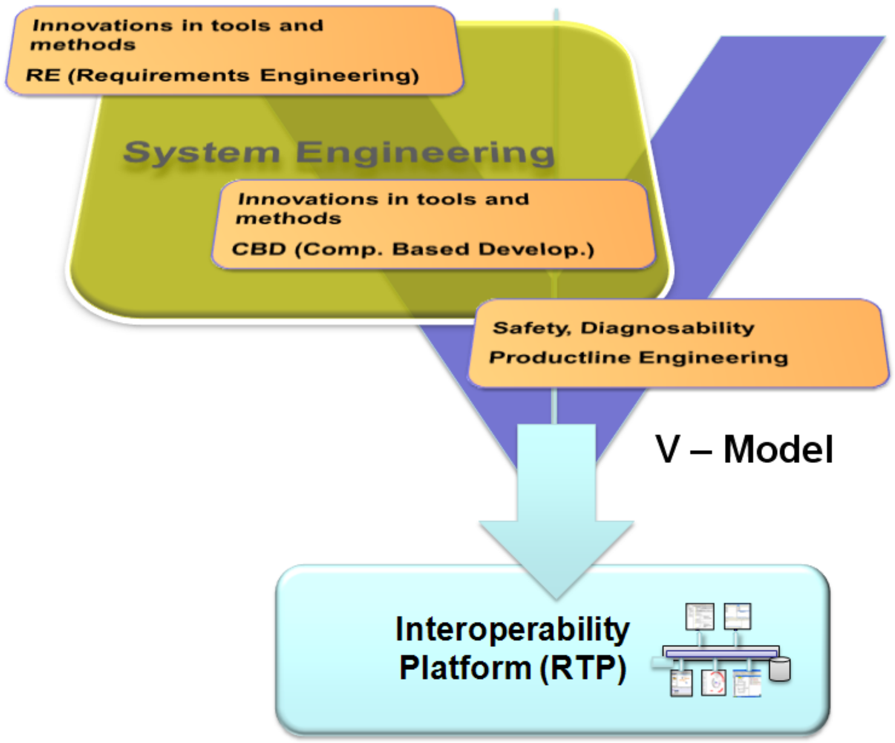

Innovations in tools and methods
RE (Requirements Engineering)
System Engineering
Innovations in tools and methods
CBD (Comp. Based Develop.)
Safety, Diagnosability
Productline Engineering
V – Model
Interoperability Platform (RTP)

Figure 1.2: The figure shows the approach of the CESAR project. Figure from [GKA+].

1.2.2 SIMOREG-plus Project

SIMOREG© [AG10] is a trade name of a SIEMENS product family for the control of DC *Power Drive Systems* (PDS). These PDS are applicable for single- and four-quadrant drives with an output range from 6.3 kW to 1900 kW. Due to hardware components discontinuation of the product family, a redevelopment is necessary. For this purpose, the project SIMOREG-plus, which is the initial trigger for this thesis, was launched to perform the redevelopment of the SIMOREG© product family. In course of this redevelopment, additional customer needs which are expressed in new functionalities have to be taken into account.

For this thesis, the existing standard stop functionalities of the PDS are of particular importance. These stop functionalities shall be extended by *certified safety-critical stop functions* to enable a direct usage of SIMOREG© DC-Masters in a safety-critical system.

1.2.3 Problem Definition and Requirements

In course of the SIMOREG-plus project, the SIMOREG© DC-Master product family shall be extended by the safety-critical stop functions *Safe Torque Off* and *Safe Stop 1* as defined in the functional safety application standard **IEC 61800-5-2** [Com05b] for *Adjustable Speed Electrical Power Drive Systems*. A conducted market analysis shows that most applications of a SIMOREG© DC-Master require a safety integrity level (SIL) of at least 2 according to the IEC 61800-5-2. The detail definition of these two safety functions within [Com05b] are:

Figure 1.3: The figure shows the SIMOREG© DC-MASTER family. Figure from [AG10].

- **Safe Torque-Off (STO):**
  Power, that can cause rotation (or motion in the case of a linear motor) is not applied to the motor. The PDS will not provide energy to the motor which can generate torque (or force in the case of a linear motor).
  - NOTE 1: This safety function corresponds to an uncontrolled stop in accordance with stop category 0 of IEC 60204-1.
  - NOTE 2: This safety function may be used where power removal is required to prevent an unexpected start up.
  - NOTE 3: In circumstances where external influences (for example, falling of suspended loads) are present, additional measures (for example, mechanical brakes) may be necessary to prevent any hazard.
  - NOTE 4: Electronic means and contactors are not adequate for protection against electric shock, and additional measures for isolation may be necessary.
- **Safe Stop 1(SS1):**
  The PDS either
  1. initiates and controls the motor deceleration rate within set limits to stop the motor and initiates the STO function (see 4.2.2.2) when the motor speed is below a specified limit; or
  2. initiates and monitors the motor deceleration rate within set limits to stop the motor and initiates the STO function when the motor speed is below a specified limit; or
  3. initiates the motor deceleration and initiates the STO function after an application specific time delay.

  NOTE: This safety function corresponds to a controlled stop in accordance with stop category 1 of IEC 60204-1.

  For the development of this PDS for DC-Motors, variant 3 of SS1 function as mentioned above shall be selected.

As mentioned in Chapter 1.1.2, it is the intention to sell the SIMOREG$^{©}$ PDS worldwide in the main automation markets. Therefore, additional safety standards have to be taken into account. A detailed description and their dependencies will be discussed in chapter 2.1. Due to the reason that such safety standards **directly impact the safety architecture**, the considered safety standards and their top-level requirements for the *safety integrity* are listed below. The investigation of the effected safety standards was already part of this thesis and is documented in the Safety Requirements Specification [Gri08].

- ISO 13849-1:
  The standard "Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design" shall be applied. According to the ISO 13849-1, the target system shall satisfy the requirements for a category **3** with a PL **d** (Performance Level).
- IEC 61508:
  The generic basis standard "Functional safety of electrical/electronic/programmable electronic safety-related systems" shall be applied. According to the IEC 61508, the component shall achieve at least SIL 2 where the E/E/PE safety-related system operates in a high-demand mode. Further details will be given in the Section Relevant Functional Safety Standards 2.1.

Based on these top-level requirements, an entire *safety concept* for a **failsafe system** shall be developed whereby the concept shall be verified by TÜV SÜD [TS11], an independent certification body. Additionally to the requirements of the relevant safety standards, the safety concept shall consider the economic facts of the SIMOREG-plus product development. Economic facts are considered in this thesis as (1) time to market, (2) product and development costs as well as the (3) experience and know-how of the department in which the system shall be developed, produced and maintained. Special attention should be paid to (1) due to the fact that the desired solution is in competition with external measures to realize safe stops in today's automation systems with DC motors and (3).

A further internal process requirement shall be an "independence" from the development of the PDS for the standard functions. In the best case, a separate internal module which extents the standard PDS with safety functions shall be developed. Therefore, a subproject in the framework of the SIMOREG-plus project has been launched which is responsible for the entire development and certification of the safety-critical system. This subproject is the main driver and coordinator for the research activities of this dissertation.

In the context of this thesis, the safety concept consists of requirements, architecture, documents and methods to ensure a safety process conformance development of the safety-critical system. In course of the development of the safety concept, different architectures shall be investigated and compared taking into account the requirements mentioned above.

1.2.4 Contribution and Significance

This research activity has been launched in the context of the SIMOREG-plus project 1.2.2. In the framework of this thesis, the entire scientific investigations of the state of the art and state of the practice of safety-critical system architectures and the corresponding methods were performed. Based on this research, this thesis claims the following two major contributions:

1. **Fail-Safe Architecture for Reconfigurable Programmable Logic Devices**
   Based on the given problem definition 1.2.3, different approaches for a suitable system architectures were developed and compared. Finally, a novel **Fail-Safe Architecture for Reconfigurable Programmable Logic Devices** (FSAR) [GFH+09, GMSW10a, GMSW10b, GSW08] was selected with the challenge to find and adapt current methods for validation and verification (V&V) to satisfy the requirements imposed by the applied safety standards. In contradiction to commonly applied safety architectures, which are mainly realized via microcontrollers, the presented solution is based on reconfigurable programmable hardware only. The significance of the developed architecture is underlined by an international patent which is valid in Austria, Europe and the United States as well as for various contributions at international IEEE conferences.

2. **Methods**
   The safety-critical developments start with the determination of the required safety integrity level (SIL), which is a result of the hazard analysis and risk assessment (H&R). A detailed application and the differences to other standards are discussed within the [GKA+, MGA+11].

   At the beginning of this thesis, the relevant safety standards did not define detailed requirements a pure hardware-related implementations built with PLDs, ASICs, FPGAs or CPLDs. The investigation of the related work and the check by the independent certification body has shown that no similar approach for a certification has been performed in the automation domain for such an architecture so far. Due to this reason, a lot of research was necessary to become confident in the development environment and the applied tool chain. Based on the novel FSAR, a new method for fault insertion testing was elaborated. A full paper describing this methodology [GMSW09] was published and presented at the IEEE International Conference for Design, Automation and Test (DATE Conference 2009).

1.2.5 Organization of the Thesis

The remainder of the thesis is organized as follows. Chapter 2, Related Work, represents the results of the related work, which is split in Relevant Functional Safety Standards, today's architectures for safety-relevant systems and analysis methods to ensure the safety evidences. In chapter 3, Safety Concept FSAR, the methodology to develop a FSAR is described. Beginning with the methods to determine the safety goals, Hazard and Risk followed by the description of the System Design, a reference workflow and a new method to perform fault insertion tests on such architectures are introduced. Chapter 4, Evaluations and Prototype, shows the application of the FSAR to an industrial prototype and the corresponding evaluation results. Chapter 5, Conclusion and Future Work, gives a conclusion of the performed work and an outlook on possible next steps and also discusses current problems with new architectures and methods in today's safety engineering. Finally, chapter 6 includes the relevant publications and the corresponding allocations to this work.

1.2.6 Mapping and Contribution of Publications

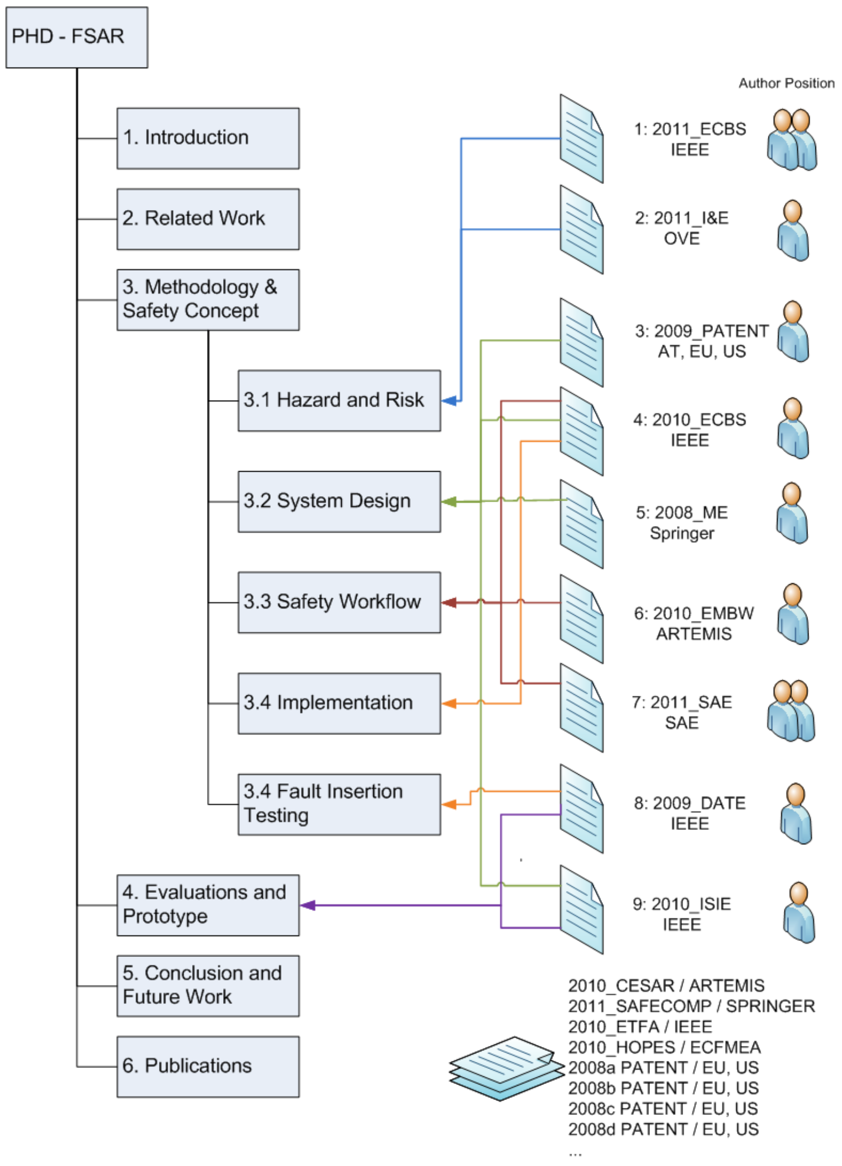


Figure 1.4: Assignment and contributions of publications

Within this section, the structure of the dissertation as well as the particular contributions and the relevance of the different publications are discussed. A "big picture" and a rough structure of this thesis are already described in "Organization of the Thesis". In addition to 1.2.5, a detailed focus is put on chapter "Safety Concept FSAR"' in which the significance and the unambiguous scientific contributions are mapped to the officially independently reviewed papers.

This thesis starts with chapter "Introduction FSAR". It reflects the evolution of the thesis and builds the overall bracket. It considers the different motivations as well as requirements derived from all listed publications in Chapter 6:

- Publication 1: A Computer-Aided Approach to Preliminary Hazard Analysis for Automotive Embedded Systems [MGA+11]
- Publication 2: Improving Automotive Embedded Systems-Engineering at European Level [GKA+]
- Publication 3: WO2009080384A1 - Method for actuating an DC machine [GFH+09]
- Publication 4: Design and Implementation of Safety Functions on a Novel CPLD-based Fail-Safe System Architecture [GMSW10b]
- Publication 5: CPLD basierende homogen redundante fehlersichere Architektur [GSW08]
- Publication 6: CESAR:Cost-efficient methods and processes for safety-relevant embedded systems [GMP+10]
- Publication 7: Model-based Toolchain for the Efficient Development of Safety-relevant Automotive Embedded Systems [AGZ+11]
- Publication 8: Fault Insertion Testing of a Novel CPLD-based Fail-Safe System [GMSW09]
- Publication 9: A CPLD-based Safety Concept for Industrial Applications [GMSW10a]

In particular, the introduction chapter starts by highlighting the needs for today's embedded systems which have to cope with challenges such as additional complexity and costs while not jeopardizing the system properties safety, reliability dependability. After an unambiguous definition of the relevant safety terms, the projects SIMOREG-plus and CESAR are described. This chapter ends with a detailed problem description, a clear statement to the **contribution of scientific significance** and the organization of this thesis.

Chapter "Related Work" contains the detailed results of the scientific investigations and comparisons with relevant work in the dedicated field of application as well as the safety state of practice. It is obvious that all publications provide a focused state of the art on the different aspects and different levels of details for this chapter.

Chapter "Safety Concept FSAR" describes the scientific significances and work performed within this thesis and published in several publications. It starts with sub-chapter "Hazard and Risk", which is a key element to enter safety-critical development. In [MGA+11] as well as in the international journalpaper of ÖVE [GKA+], the approach of an *Hazard Analysis and Risk Assessment* (H&R) is deeply discussed. Furthermore, the difference between the determination of the safety integrity level between the generic basic standard IEC 61508 [Com02] and the automotive application standard ISO 26262 [Int10] are described.

After the H&R, the focus is put on sub chapter "System Design" on the fail-safe system architecture and its features, the, patent [GFH+09] valid in the US, EU and AT underlining the significance of the chosen fail-safe architecture. The scientific relevance is given by the publications of this architecture at the international conferences (IEEE, Springer) [GMSW10b, GSW08].

Sub-chapter "Safety Workflow" depicts the applied tool chain [GMSW10b] for the particular problem definition. In context with publications [GMP+10, AGZ+11], the strong need for an integrated tool-chain as well as the seamless safety argumentations for such tool chains are clearly expressed.

Sub-chapter "Implementation" with publication [GMSW10b] shows the realization of the required safety functions according to the IEC 61800-5-2 as well as the implemented measured to ensure the adequate safety integrity according to IEC 61508 and ISO 13849.

In sub-chapter "Fault-Insertion Testing", a new methodology for fault-insertion testing was developed. The developed method for "**Fault-Insertion Testing of a Novel CPLD-based Fail-Safe System**" [GMSW09] underlines the significance of this work and was published and presented on DATE Conference 2009.

The main part for chapter "Evaluations and Prototype" are contributed by publications [GMSW09, GMSW10a]. In [GMSW10a], which is a publication at the largest industrial IEEE summer conference yearly, an application of the fail-safe architecture was described. Furthermore, evaluation results from fault insertion testing from the DATE publication [GMSW09] are discussed.

Chapter 2

Related Work

The picture 2.1 shows an overview of the number and the year of references listed in the bibliography at the end of this thesis. It reflects the two phases of the related work analysis at the beginning (peak 2008) and the end (peak 2010) of this thesis.

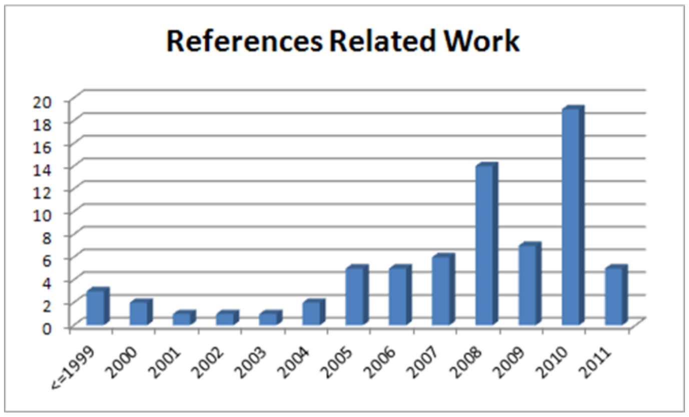


Figure 2.1: Overview references in the literature chapter (Axes:Number/Year)

2.1 Relevant Functional Safety Standards

As discussed in section 1.1.1, safety is defined as the ability of a system to not cause or contribute to the occurrence or aggravation of harm. In this context, the overall goal associated with safety is to transform the severity and likelihood of risk inherent in all human activity to lower, acceptable levels.
Therefore, independent from the domains, functional safety standards deal with product safety or system safety. In particular, they prescribe the application of special technical and managerial skills and procedures in a systematic, forward-looking manner in order to identify and control hazards throughout the entire life cycle of a project program or activity.

Of particular interest for this work are safety standards in the field of automation for **machinery**. An overview of the relevant standard landscape is described in [AG07]. According to the "Hierarchy of European standards for machine safety", the European safety standards can be split in (1) basic safety standards (type A standards), (2) safety group standards (type B1 standards and type B2 standards) and (3) machine-specific technical standards (type C standards).

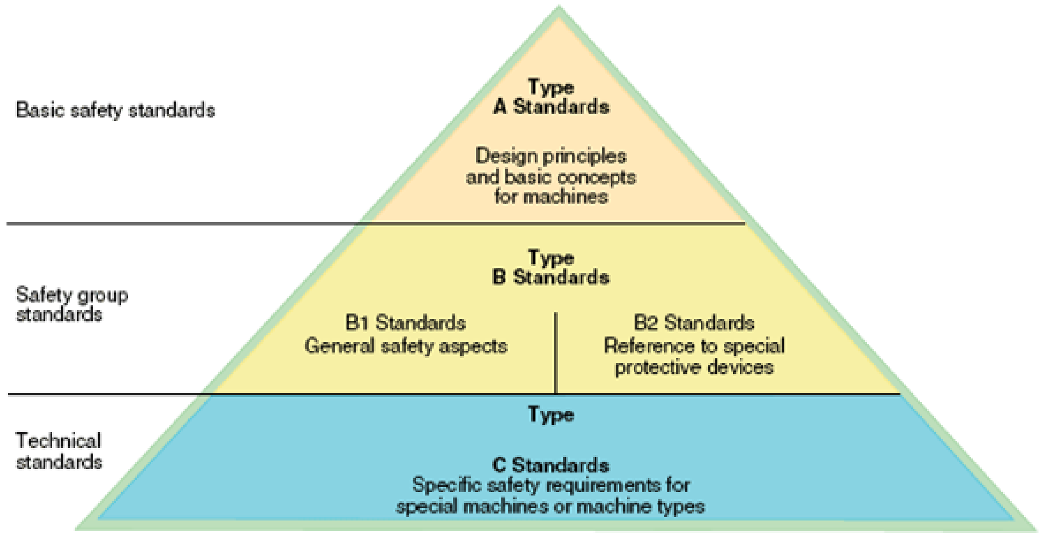


Figure 2.2: Hierarchy of European standards of machinery safety. Figure from Lueze Electronics.

- **Type A Standards**
  The design principles and basic concepts defined in type A standards are mandatory for all machines. Examples are EN ISO 12100-1 “Safety of machinery - Basic concepts, general principles for design” or ISO 14121-1/EN 1050 “Safety of machinery - Principles of risk assessment” . This kind of standard provides (1) methods and instructions for determining risks 1.1.1 that are connected with the machine and (2) different approaches for preventing risks with the objective of integrating safety, even before the machine manufacturing begins. Also the generic safety standard IEC 61508 [Com02] can be classified as a kind of Type A Standard [LPP10].

- **Type B Standards**
  Type B Standards are divided into (1) Type B1 standards where generally higher safety aspects with possible solutions for these aspects are described, such as ISO EN 13849 [fS06] “Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design”, and (2) Type B2 standards where normative requirements of special protective devices, such as emergency STOP buttons (e.g. IEC 60204-1 [Com05a]), safety door switches etc. are grouped together. The manufacturer of such products and the machine designer must consider the requirements of these standards for the design and testing of safety components in the usage in their machines.

- **Type C standards**
  This type of standard, also called sector standard, describes specific measures for

reducing risks in a particular field of application, special machines or machine types. If a C standard exists for the machine type in question, it takes **priority over a B type or A type** standard defined above. If there are additional hazards that are not addressed in the standard, or if there is no special C standard for the machine being planned, risk reduction in accordance with A and B standards must be made. The IEC 61800-5-2 [Com05b] is a typical type C standard.

2.1.1 IEC 61508

The first relevant standard is the application-independent standard for functional safety. This international standard sets out a generic approach for all safety lifecycle activities for systems comprised of electrical and/or electronic and/or programmable electronic (E/E/PE) elements that are used to perform safety functions [Bör07]. The first edition of the IEC 61508 [Com02], which was published in the year 2000, does not define special requirements for the development of safety-related systems using CPLDs. The responsible committee for the second edition of the IEC 61508 has detected this gap. Thus, the second edition of the IEC 61508 [Com10], which replaced the first edition in April 2010, includes detailed requirements concerning the development of safety functions using ASICs, FPGAs and CPLDs.

During the development phase of the FSAR, a draft of part 2 of IEC 61508 was available, which includes requirements for design, implementation and verification of CPLD-based safety-critical systems. The draft defines a V-model of the safety lifecycle for ASIC, FPGA and PLD designs. There are similarities between the development of safety-critical software and the development of ASIC, FPGA and PLDs. Thus, the V-model of ASIC, FPGA and PLD designs is similar to the V-model of the development of safety-critical software. Additionally, Annex B of the draft of part 2 contains a table which defines measures and techniques to avoid systematic faults in the development process of user-programmable (PLD, CPLD and FPGA) for different safety integrity levels. There are requirements for design entry, synthesis, placement, routing, layout generation and production. The presented fail-safe system to experimentally evaluate the FSAR concept was developed in adherence to the "available draft 2008" of part 2 of IEC 61508. Additionally, guidelines (e.g. [SHM05]) for developing safety-critical systems using hardware description languages were considered. With [Cle09] a paper is published which identifies how faults/failures arise in FPGA and which mitigations can be applied in the development. This knowledge serves as basis for an approach to arguing FPGA safety as well as a framework for safety assessments with IEC 61508 standard.

The IEC 61508 introduces definitions and terms to enable a systematic approach to compare and evaluate the safety integrity of different E/E/EP systems. Similar definitions for other safety standards are derived from the basic standard IEC 61508 like the ASIL (Automotive SIL) in ISO 26262 [Int10]. The proof of the safety integrity is one of the most important topics in the independent certification or qualification of safety-critical systems. Finally, one major goal is the achievement of a SIL 2 for the FSAR according to IEC 61508 whereby the SIL is directly connected to the **Safe Failure Fraction** (SFF) 2.1 and the **operation mode**. The relevant operation mode is already predefined as **continuous mode** in the IEC 61800-5-2 in section 2.1.3. To be able to determine the SFF, respectively the SIL, further definitions are necessary. In the following, some important definitions from

SIL	Probability of a dangerous failure per hour (PFH_{sys})
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

Table 2.1: Safety Integrity Level (SIL) in high demand or continuous mode of operation.

the IEC 61508 are listed.

- **Safety Integrity**
  Is the probability of an E/E/PE safety-related system satisfactorily performing the required specified safety functions under all the stated conditions within a stated period of time.

- **Safety Integrity Level(SIL)**
  Is the discrete level (one out of a possible four), corresponding to a range of safety integrity values, for specifying the safety integrity requirements of the safety functions to be allocated to the E/E/PE safety-related systems, where safety integrity level 4 has the highest level of safety integrity and safety integrity level 1 has the lowest, see table 2.1.

- **Hardware Fault Tolerance(HFT)**
  The HFT (Hardware Fault Tolerance) of a safety-critical system is determined by the number of failures which can cause a loss of one or more safety functions. In particular, a system with an HFT of N means that $N+1$ faults can cause the loss of the safety function. Due to this reason, systems with an HFT of N consist of $N+1$ channels which have the capability to independently carry out the safety functions.

- **Failure Rate λ**
  safety-critical systems consist of one or more components, whereby each of these components has its own failure rate λ. Due to the "aging", the failure rate is changing over the time, but for a specific mission time the failure rate can be simplified and assumed to be constant. The corresponding failure rates will be determined in an experimental manner only. The failure rated is split in:

  - Rate of safe detected failure λ_S (S ... Safe)
  - Rate of detected dangerous failures λ_{DD} (DD... Dangerous Detectable)
  - Rate of undetected dangerous failures λ_{DU} (DU... Dangerous Undetectable)

- **Safe Failure Fraction (SFF)**
  The SFF of a safety-critical system describes the ratio of the average rate of safe failures plus detected dangerous failures of the E/E/PE safety-related systems towards the total average failure rate of that system.

$$SFF = \frac{\sum \lambda_S + \sum \lambda_{DD}}{\sum \lambda_S + \sum \lambda_{DD} + \sum \lambda_{DU}} \tag{2.1}$$

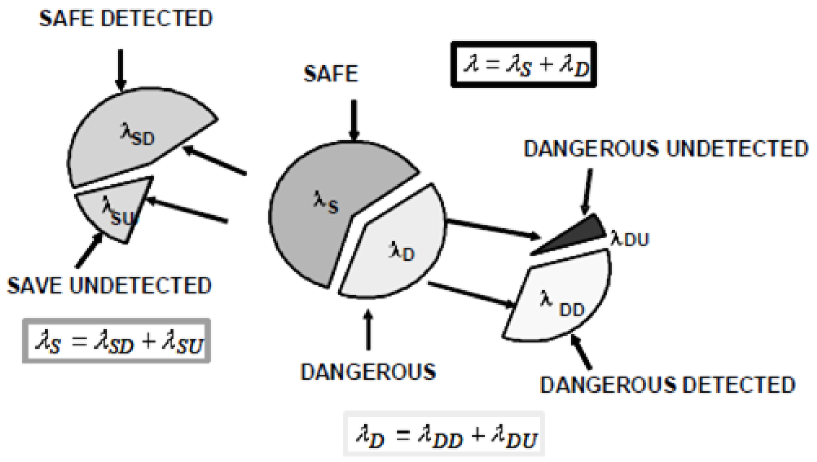


Figure 2.3: Failure rate distribution. Figure from [BHU08].

In the position paper of EXIDA [EXI10], a redefinition of this quantity is proposed. Failures like *fail annunciation*, *failures with no effect* or *residual failures* shall not be included in the SFF calculation in order to not falsify the SFF.

- **Diagnostic Coverage** (*DC*)
  The *DC* (Diagnostic Coverage) is the fraction of dangerous failures detected by automatic on-line diagnostic tests. The fraction of dangerous failures is calculated using the dangerous failure rates associated to the detected dangerous failures divided by the total rate of dangerous failures.

$$DC = \frac{\sum \lambda_{DD}}{\sum \lambda_{DD} + \sum \lambda_{DU}} \tag{2.2}$$

- **Common Cause Failure** (*CCF*)
  (*CCF*) are failures that are the result of one or more common events causing coincident failures of two or more separate channels in a multiple channel system leading to system failure. The probability of a common cause failure will be expressed by the β factor. Therefore, two kinds of beta factors are defined:
  - Rate of undetected system failures in case of common cause β
  - Rate of detected system failures in case of common cause β_D

2.1.2 EN ISO 13849

The second basic standard concerns the safety of machinery and is mainly relevant to vendors of safety-critical components for machinery. The first part of the EN ISO 13849-1 [fS06] provides safety requirements and guidance on the principles for the design and integration of safety-related parts of control systems, including the design of software. This standard defines so-called "categories" (Cat) and "performance-levels" (PL) to determine the safety integrity.

As mentioned in section 1.2.3, the aim of FSAR is to achieve a category 3 with a performance-level d to make the presented architecture applicable on our target market without any limitations. Consequently, two requirements defined in EN ISO 13849-1 concerning the system architecture and the required safety integrity need to be fulfilled:

- A single fault must not cause the loss of the safety functions.
- A single fault has to be detected at the time when a safety function is demanded or earlier.

These requirements constrain the use of an architecture consisting of two independent channels. **Thus, the required hardware fault-tolerance (HFT) is 1.** It is not possible to satisfy these requirements with an architecture consisting of a single channel.

2.1.3 IEC 61800-5-2

The IEC 61800-5-2 [Com05b] is a derived sector standard from the generic basic standard IEC 61508. Generally, derived sector standards are describing requirements for special fields of application. In this work, the international standard IEC 61800-5-2, which describes the requirements for adjustable speed electric drive systems, has to be considered for the FSAR as well. As mentioned above, the 61800-5-2 is a typical C standard and has priority over the other safety standards listed above.

This International Standard is only applicable if functional safety of a power drive system (PDS) is claimed and the PDS (safety-related) is operating in the **high-demand** or **continuous mode**. Figure 2.4 shows the functional elements of a PDS that are considered in this part of IEC 61800-5-2.

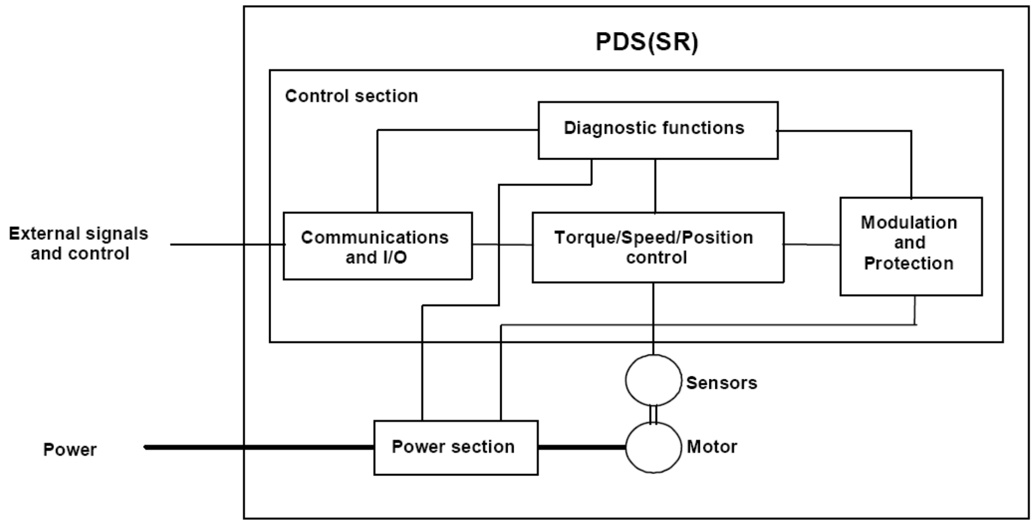


Figure 2.4: Considered parts in IEC 61800-5-2 of a PDS (safety-related). Figure from IEC 61800-5-2.

2.2 Safety Architectures

Various architectures for safety-critical systems have been proposed so far. In general, safety-critical system architectures shall provide a deterministic behavior under conditions of failure. In [SD06], architectures and strategies to achieve safety integrity are described taking into account several criteria for the selection of the suitable architecture whereby the typical criteria for selecting a specific strategy are split in:

- **Primary criteria**
  The selected architecture must meet system safety requirements. E.g. failure management within the system safe response time (Meaning the system transition to a safe state within the required safe fault response time).

- **Secondary criteria**
  1. Level of independent checking is provided
  2. Performance
  3. Available technology
  4. Effort of development

Micro-controllers are frequently used for the realization of safety-critical embedded systems [SD06]. A drawback is that they require the implementation of complex safety integrity measures like RAM and CPU tests. The implementation of these safety integrity measures requires large efforts. If the safety functions, which need to be realized, are comparably incomplex, the largest part of the implemented source code is dedicated to the testing of RAM, CPU and periphery, and only a little source code is dedicated to the implementation of safety functions. In the following, different approaches for fail-safe controller architectures will be discussed, highlighting the benefits of each approach.

- **Single-controller Strategy**
  A single-controller failsafe strategy, as shown in figure 2.5, consists of a single micro-controller executing the target application software and self-tests which collaborate with a primitive watchdog. The primitive watchdog provides a limited-level of independent checking for the micro-controller. The basic idea is that the micro-controller has to periodically trigger the watchdog. If such a trigger is not performed within a defined time interval, the watchdog resets the micro-controller. The disadvantage of such an architecture is that a simple watchdog has no information about the internal calculation state or potential failures of the micro-controller, e.g. an incorrect output calculation.

- **Symmetric-controller Strategy**
  Two identical micro-controllers (homogeneous redundancy) execute the same program, and their computations depend on the same inputs. The controllers communicate and compare their results in order to detect faults. Additionally, they perform checks for communication time-outs. Thus, each micro-controller serves as watchdog for the other micro-controller. This strategy has the advantage to provide a good diagnostic coverage of hardware faults and random hardware failures with minimal

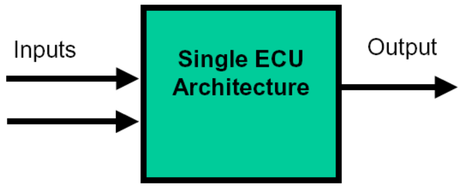


Figure 2.5: Single-controller strategy consists of a micro-controller which collaborates with a primitive watchdog. Figure from [SD06].

effort devoted to developing self-checking diagnostics. This strategy has to cope with challenges such as (1) in the synchronization of the micro-controller, (2) real-time application and (3) the necessary consideration of common cause failures, see section 2.1.1.

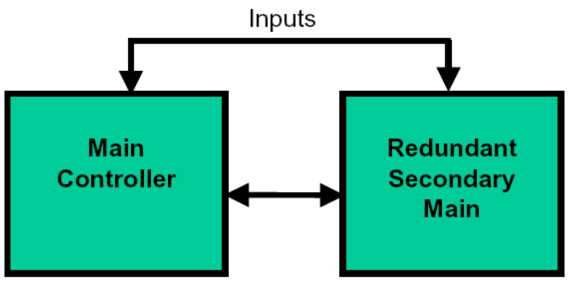


Figure 2.6: Symmetric-controller strategy consists of two identical micro-controllers (homogeneous redundancy). Figure from [SD06].

- **Asymmetric-controller Strategy**
  An intelligent watchdog (ASIC or low cost processor) collaborates with a micro-controller. The intelligent watchdog verifies the micro-controller's integrity by requesting periodic diagnostic checks. The strategy has the potential to provide a high level of independently checking at lower costs compared to the symmetric processor strategy.

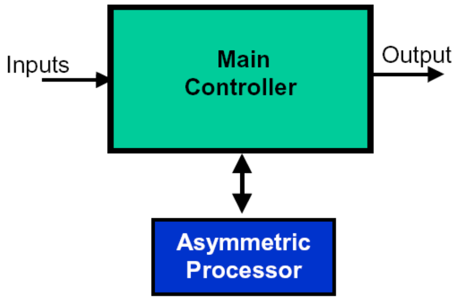


Figure 2.7: Asymmetric-controller strategy consists of a micro-controller and an intelligent watchdog (ASIC or low cost processor). Figure from [SD06].

The asymmetric controller strategy is currently the most frequently applied strategy in the automotive industry. Based on the basis EGAS concept [ABD+07] by Robert Bosch, the EGAS Working-Group has developed a standardized SW and HW concept to control and monitor the torque in gasoline and diesel engines. Today, the concept is used in many different applications. The basic idea is a diverse calculation of the safety functions in the function controller with an independent intelligent check of the controller commands by a monitoring unit 2.8.

However, the asymmetric controller strategy has the same disadvantage as the single-controller strategy due to the reason that no information about internal calculation state or potential failures from the function controller are available on the monitoring unit.

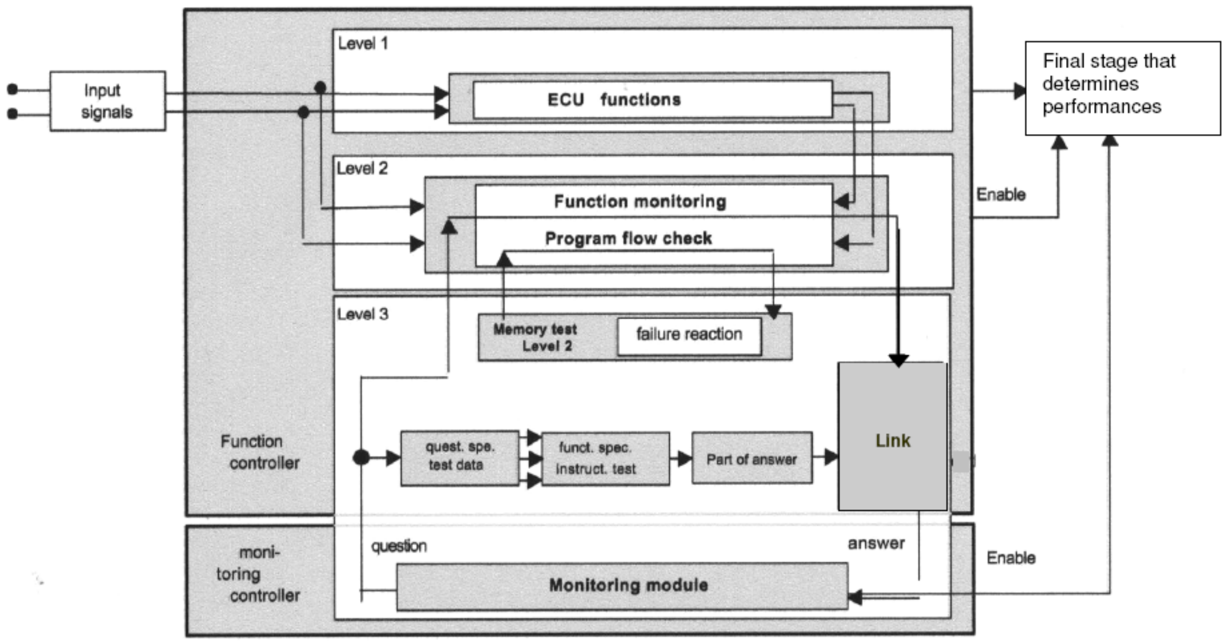


Figure 2.8: EGAS monitoring concept for engine management systems of gasoline and diesel engines. Figure form [ABD+07].

- **Dual-core controller Strategy**
  The basic principle of a dual-core strategy is the duplication of the core in one microchip. Due to this duplication and a close communication between the cores, e.g. *lock step mode*, this strategy is able to easily detect discrepancies. The dual-core approach provides a high safety integrity and a high coverage of random hardware failures. In [RW10], the applicability of a multi-core technology in the light of [Com02] is analyzed. The different components (hardware, software and operation system) of the multi-core architecture are evaluated concerning their redundancy, diagnostic coverage and diversity. Furthermore, commercially available multi cores are compared. Dual-core and multi-core architectures are very promising approaches,

but the major disadvantage of such strategies are the $\beta - factor$ and the associated common cause failures, see section 2.1.1.

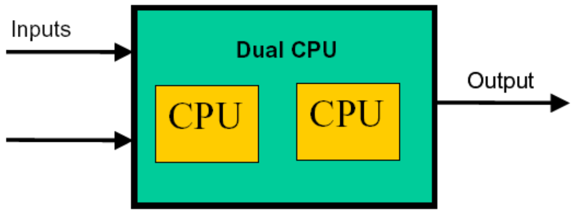


Figure 2.9: Dual-core controller strategy. Figure from [SD06].

- **Distributed controller Strategy**
  The principle of the distributed controller strategy is similar to the symmetric controller strategy where the safety function is calculated two times in two independent controllers and the results are be compared afterwards. Additionally to the symmetric controller strategy this concept has further challenges like available network bandwidth, fault response times, available controller throughput, synchronization issues etc. which have to be taken into consideration prior of the selection of this approach. This strategy sounds very promising for an application in vehicles. In order to find solutions to those challenges mentioned above the ARTEMIS R&D project Pollux [Pol10] was launched in 2010 with the objective to develop a distributed real time embedded systems platform for the next generation electric vehicles.

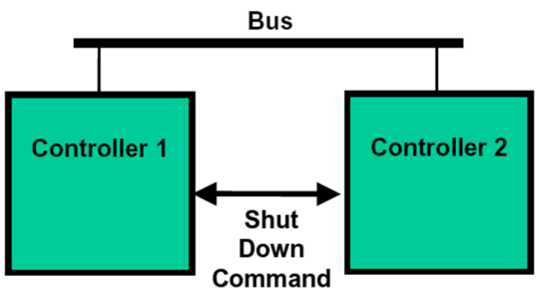


Figure 2.10: Distributed controller strategy of two micro-controllers connected via a bus. Figure from [SD06].

In [BHU08], safety systems on a single FPGA are discussed. The authors introduced a 1oo2-RISC architecture with a controller unit which compares their results, in case that both processors providing different results the system enters a safe state. A focus is put on placement and routing of the components on the FPGA as well as the tools supporting these. Furthermore, a first evaluation of the safety characteristic (e.g. PFD, failure rate) is performed taking into account the [Com02]. However, the CCF, which is the limiting factor to apply and certify such architectures, is not treated within this work.

The controller unit mentioned in [BHU08] is realized in [ZAMY10] in a separated FPGA for a 2oo4-system. The micro-controller data are compared between the depended micro-controllers to detect faults. Additionally, the controlling of the data storage the FPGA is intended to connected micro-controllers to the SRAMs. Furthermore, the data are extended with an error detection code (hamming code) to increase the reliability of the system. The diversity of this approach is very promising due to the fact that different architectures (micro-controllers and FPGAs) are connected, however, there are no statements on the safety-related process for the FPGA development and the problematic of the CCF for FPGA, which is the single element of this distributed architecture.

In [BFM$^+$03], fault-tolerant architectures are reviewed. The authors investigated how to implement architectures, which contain multiple CPUs on a single chip. The considered architectures are lock-step, loosely-synchronized dual-processor and triple modular redundant (TMR). Additionally, the authors propose new architectures for the integration into a single chip only. The integration of fault-tolerant systems into a chip is problematic because of common cause failures. These failures affect all duplicates of a circuit due to the same cause (e.g. faulty clock tree or power supply). Thus, it is difficult to reach high levels of safety integrity with this approach without additional measures.

Although the integration of safety-critical systems into a single chip is difficult, an approach to reduce the probability of common cause failures is described in [MF07]. The authors use a library of Intellectual Property blocks, which can be used for fault detection and fault-tolerance. Their architecture includes a block which performs memory protection. A supervisor performs diagnostic checks of the CPU. The checks are mainly implemented in hardware. The system bus and the interfaces to peripheral components are supervised by separate blocks. Another bus system is used for the communication between the diagnostic units. The authors refer to this approach as faultRobust.

The approach reduces the probability of the occurrence of a common cause failure, as some blocks are architecturally or functionally diverse. Thus higher levels of safety integrity can be realized without additional measures to decrease the probability for common cause failures. To certify intellectual property, which belongs to the faultRobust approach, in adherence to the standard IEC 61508, a system FMEA (failure modes and effects analysis) on SoC (System on Chip) level can be used, as described in [MBC07]. With this methodology, it is possible to assess the SFF (safe failure fraction) of the SoC.

A TMR (triple modular redundancy) system consisting of multiple PLDs is described. In [AMF05], the utilization of such PLDs to design safety-critical systems is proposed. The authors claim that most of the faults which affect safety-critical systems occur at the interfaces. Thus, the integration of safe input cells and safe output cells into the PLDs is proposed.

A safe input cell receives inputs from three sensors which measure the same safety-critical physical quantity. A voter circuit determines the probably correct value of the physical quantity. Safe output cells represent the logic value 1 by a periodic signal, while the logic value 0 is represented by a constant signal. Safe input cells and safe output cells comprise an additional state machine, circuitry for the detection of failures and circuitry for the signaling of errors to other system components.

Totally self-checking circuits are described in [BMSS00]. The output of circuits is separated into code words and noncode words. If no fault occurred, the output of a circuit is a code word. If the output of a circuit is a noncode word, a fault occurred. A dedicated circuit (checker) decides whether the output of a circuit is a code word or a noncode word and signals a fault if necessary.

The approach reduces the probability of the occurrence of a common cause failure as some blocks are architecturally or functionally diverse. Thus, higher levels of safety integrity can be realized without additional measures to decrease the probability for common cause failures.

The authors of [MSM00] discuss common mode failures (CMF) in redundant systems focusing on VLSI systems. They refer to the term common mode failure as the result of an event which, because of dependencies, causes a coincidence of failure states of components in two or more separate channels of a redundancy system, leading to the defined system failing to perform its intended function. The authors use the terms "common mode failure" and "common cause failure" interchangeably.

According to the authors, CCF can occur in hardware or software. They can be caused by permanent faults (e.g. bugs) or intermittent faults (e.g. weak signals) introduced during the development process of a redundant system. Exemplary faults are ambiguous specifications, bugs of used tools, incomplete verification, manufacturing defects or non-exhaustive testing. CCF can also be caused by external disturbances during the operation of the redundant system. Caused effects can reside transiently or permanently.

Techniques to handle common cause failures comprise CMF avoidance (use of mature and verified components, conformance to standards, use of formal methods, use of design automation, implementation of design rules, diverse realization of redundant components), CMF removal (design reviews, simulation, verification, testing and fault-insertion) and CMF tolerance (watchdog timers, exception handlers, runtime checks, concurrent error detection).

2.3 Safety-Critical Methods

As described in the IEC 61508 and mentioned in section 1.1.1, the two major aims of functional safety standards in the development phase are (1) the avoidance and control of systematic errors during the development and (2) the control of malfunctions and random hardware failures during the operation.

To support the first aim mentioned above, various categories of safety and dependability analyses can be identified, corresponding to various categories of objectives, depending on the system to be developed. Safety is a subject across the different engineering phases, implying various approaches depending on the development phase addressed. Early stages of the safety life-cycle aim at the identification of requirements and the exploration of the implications of design, whereas later life-cycle stages focus on the successful implementation of the requirements.

Typical safety analyses required by safety standards include:

- Hazard analysis and risk assessment
- System Safety Assessment, generally supported by Fault Tree Analysis, Failure Modes & Effects (and Criticality) Analysis (FMEA), Common Cause Analysis etc. [ZLQ10]
- Verification and validation (e.g. fault insertion tests), in particular of the implementation of safety and functional safety requirements

In [CB10], methods are introduced to analyze a modular design embedded in an FPGA exhaustively. In particular, a semi-automated FPTC (Fault Propagation and Transformation Calculus) analysis technique is used to perform safety analysis and to support the creation of the safety case. This bottom-up approach can be linked with other safety analyses like a FMEA in order to enhance and verify safety properties required.

An example to design and test FPGA-based safety-critical systems is introduced in [KSS09]. This paper describes regulatory requirements for safety-critical systems as well as methods and test-applied in different development stages of FPGA projects in nuclear power plants. The authors investigate the fulfillment of the safety requirements on a low-level. Complementary to this detailed low level analysis of a single FPGA and its components (e.g. lock-up tables), this thesis focuses on arguing safety on system level (FSAR).

One important method in the domain of industrial automation is *fault insertion testing*, which is usually applied for the verification of micro-controller-based fail-safe systems. The aim of fault insertion tests is to emulate faults in the system hardware [Com02]. Then the responses of the faulty system are analyzed. These kinds of tests are used to assess the dependability of the system in case of a fault.

A fault can be injected into a micro-controller by modifying the executed program, which is usually implemented using C or C++. In this case, the program can be modified with preprocessor commands to inject a fault. Then, the program needs to be recompiled and loaded into the micro-controller, which has to emulate the fault.

In contrast to micro-controller-based fail-safe systems, the fail-safe system based on the FSAR requires a new method to perform fault insertion testing because these devices do not execute programs implemented in C or C++.

In [GBGG01], VHDL-based fault insertion techniques are discussed. A saboteur is a component that is added to a VHDL-description to alter the timing characteristic or the value of a signal if the saboteur is activated. During the normal operation of the system, the saboteur is inactive. Saboteurs can be used to inject various fault types like stuck-at faults, bit-flips, bridging-faults or delay faults. Mutants are components which replace corresponding components. If a mutant is activated, it behaves like the corresponding component in presence of a fault. If the mutant is inactive, it behaves like the fault-free corresponding component. A mutant can either be created by adding saboteurs to a structural model description, by replacing subcomponents of a structural model description or by modifying the syntactical structures of a behavioral description [JAR$^+$94]. It is possible to inject various fault types like assignment control, stuck-else or stuck-then using mutants [GBGG01].

In [BHU08], a fault insertion is mentioned in a 1oo2-RISC FPGA architecture by manipulating the VHDL-code of one RISC processor. An additional controller compares signals from both RISC processors and generates an alarm in case of inconsistency.

In [GYJ97], the use of PLDs is proposed to accelerate a fault simulation. A PLD is connected to a host computer. The circuit which has to be simulated is entirely or partly mapped on the PLD. The host computer executes a simulation program, which applies input vectors to the circuit. The responses are read back by the host computer. The authors distinguish between dynamic and static fault-insertion. In case of dynamic fault-insertion faults are injected during run-time. Multiplexers are inserted into the circuit to emulate stuck-at faults. The multiplexers contain two inputs. One input is connected to the correct input value and the other input is connected to a 1 or a 0 (stuck-at fault). The host computer controls the selector wire of each multiplexer. Thus, it can determine if the correct input value or the stuck-at fault is switched to the output of the multiplexer. The advantage of this approach is that the circuit description has to be compiled only once. Also, the PLD has to be configured only once. The disadvantage is that the multiplexers require additional PLD resources. Furthermore, the delay of the circuit increases, limiting the maximal clock speed. An alternative approach is static fault-insertion. In this case faults are injected into the circuit during compile-time. Thus, whenever the circuit has to be simulated using another set of faults, the circuit description needs to be recompiled and the PLD needs to be reconfigured. The disadvantage is that the frequent recompilation and reconfiguration increases the duration of the simulation. The advantage is that no additional multiplexers are inserted. Hence, no additional PLD resources are needed and the delay of the circuit is not increased.

In [ERTU06], the use of a single FPGA as fast simulation environment is described. A faulty version of a circuit and a fault-free version of the same circuit are emulated concurrently. A comparator compares the output of the faulty circuit to the output of the fault-free circuit. If the output differ from each other, a fault was detected. An LFSR (linear feedback shift register) is used to generate test vectors for the circuits. An additional state machine controls the fault simulation.

The IEC 61508 requires fault insertion testing for the verification of safety-related systems. Thus, fault insertion testing is an important prerequisite for the successful certification of a system in adherence to the IEC 61508.

Chapter 3

Safety Concept FSAR

There are a lot of interpretations in the different standards for the term *Safety Concept*. For instance, the ISO 26262 [Int10] splits the functional safety-related activities in the concept phase in

1. Item definition,
2. Initiation of the safety lifecycle,
3. Hazard analysis and risk assessment (H&R),
4. Functional safety concept.

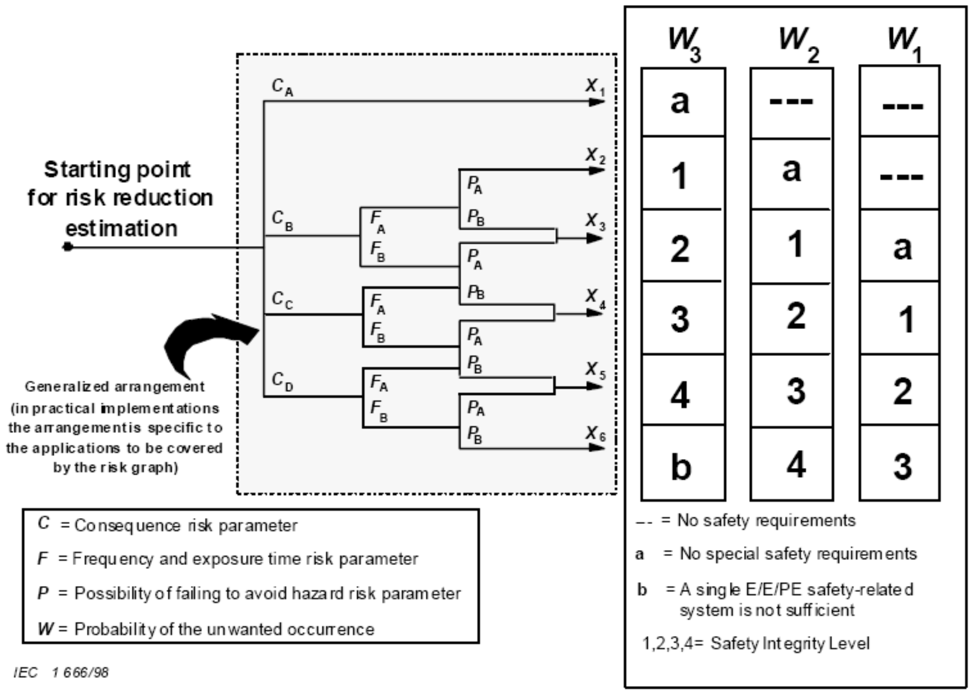
Within this thesis, the term **Safety Concept** will be used for all applied methods, tools and activities including the safety architecture with its safety integrity measures. The aim is to ensure a seamless safety argumentation chain for the development of the safety-related product. Here, term *Safety Concept* refers to an extended concept phase which additionally includes a detailed technical safety concept, System FMEA, V&V Plan and the "qualification" of the used tool chain. In contradiction to the *Safety Case*, for which the focus is on providing traceable safety argumentation for all activities, the focus of the *Safety Concept* is on planning the development and V&V of a safety-related product. The *Safety Concept* phase in this case ends with an official *Proof of Concept* from an independent assessment institute such as TÜV-Süd. In the following sections the main activities will be discussed.

3.1 Hazard and Risk

The determination of the requested safety integrity as well as the term describing the safety integrity ase slightly different for the various functional safety standards.

In the automotive domain, a necessary first step in the process for the development of a new vehicle is the application of **hazard analysis and risk assessment (H&R)** [MGA$^+$11,JWR07] by a team of people with a wide variety of knowledge and skills [Nan95]. This analysis technique is applied qualitatively before concrete design solutions are elaborated and enough numerical values are defined to allow the application of other techniques such as simulation or quantitative safety analyses. The purpose of the early application of hazard analysis and risk assessment are the identification and assessment of potential hazards of a newly developed vehicle that are caused by potential failures of the vehicle's embedded system.

The early knowledge about the existence of hazards allows the definition of safety goals (top-level safety requirements to the embedded system) with the desired ASILs, even if detailed and quantitative information about the vehicle under development is insufficiently available. Considering the safety goals, a safety-critical embedded system design (typically including runtime fault detection capabilities) is developed that is able to best achieve the defined safety goals and to control or mitigate the identified hazards. Although the application of H&R alone cannot ensure safety of a vehicle, it is a necessary first step in order to eliminate or control hazards through adequate design, implementation, integration, verification and validation. The early application of a hazard analysis is required by the safety standard ISO 26262 for automotive E/E system development.


Starting point for risk reduction estimation
Generalized arrangement (in practical implementations the arrangement is specific to the applications to be covered by the risk graph)
C_A
C_B
C_C
C_D
F_A
F_B
P_A
P_B
X_1
X_2
X_3
X_4
X_5
X_6
W_3
W_2
W_1
a
1
2
3
4
b

C = Consequence risk parameter
F = Frequency and exposure time risk parameter
P = Possibility of failing to avoid hazard risk parameter
W = Probability of the unwanted occurrence
--- = No safety requirements
a = No special safety requirements
b = A single E/E/PE safety-related system is not sufficient
1,2,3,4= Safety Integrity Level
IEC 1 666/98

Figure 3.1: SIL Determination according to IEC 61508-5. Figure from [Com10].

In the automation domain [GSW08], the control of hazards and the determination of the safety integrity level follows the same principle. In particular, the IEC 61508, as application-independent basic standard, also includes a mandatory hazard and risk assessment in order to identify the adequate safety integrity level (SIL) as a combination of (1) consequence, (2) exposure time, (3) possibility of failing to avoid hazard and (4) probability of the unwanted occurrence. The determined SIL directly impacts the entire development process of the safety-related product. One noteworthy difference in the H&R between automotive standard (ISO 26262) and automation standard (IEC 61508) is that in the ISO 26262, the effects of hazards are assessed in the context of possible operational situations, deriving a corresponding ASIL, which is a combination of (a) severity, (b) exposure and (c) controlability. Contrary to the IEC 61508 where factor (3) can reduce the required SIL one level only, the comparable factor (c) in the ISO 26262 can fully impact the derived ASIL as discussed in [GKA+]. In case the hazard is controlable by the driver, no ASIL independent from severity or exposure is required.

It is obvious that the determination of the corresponding safety integrity level strongly depends on the target application. However, in case of product (components) development, like the SIMOREG-plus project, the detailed target applications are not known. Therefore, only a rough determination of the SIL can be performed. In this case, the determination is based on (1) experience from the target market including its requirements, (2) possible applications with a rough SIL determination and (3) potential new economic facts. The results of this SIL determination and the derived safety requirements are described in the Safety Requirements Specification [Gri08] and listed in section 1.2.3.

3.2 System Design

3.2.1 Fail-Safe Architecture

The presented fail-safe system is based on the novel CPLD-based fail-safe system architecture which was patented by SIEMENS [GFH+09] in 2008 and first time presented in [GSW08]. The fail-safe system contains two channels (HFT=1). Each channel is able to perform the safety functions independently. Thus, if a channel is affected by a fault, the other channel is still able to execute the safety functions. The entire system enters the safe state if a faulty channel is detected. Figure 3.2 illustrates the fail-safe system. It comprises the following entities:

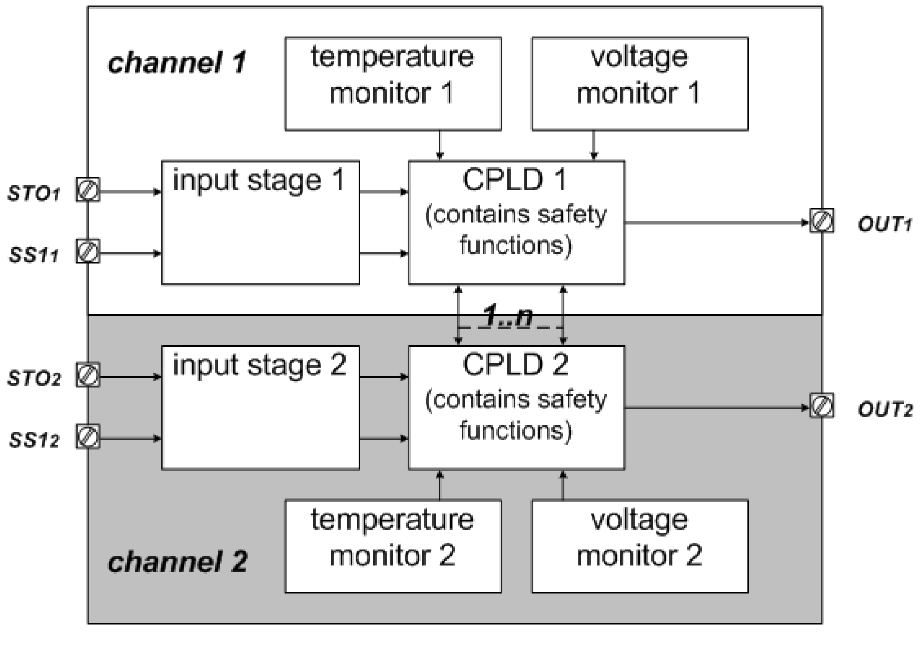


Figure 3.2: Fail-safe System Architecture (see figure 1.4.

- The terminals **STO**$_1$, **SS1**$_1$, **STO**$_2$, **SS1**$_2$ are connected to an external device (e.g. a control panel) that can be used to activate the safety functions STO and SS1.
- The **input stages** represent an interface for the activation of the safety functions that assures electrical isolation by optocouplers.
- The two **CPLDs** realize the safety functions. In order to be able to perform diagnostic checks, the CPLDs exchange signals. The CPLDs control the safety-critical outputs (OUT$_1$ and OUT$_2$) of the fail-safe system. Each CPLD is clocked by a separate external oscillator.
- The terminals **OUT**$_1$ and **OUT**$_2$ are connected to an electric motor to allow power to be applied to the motor or to respectively shut down the motor.
- Each channel contains a **temperature monitor** to detect temperature deviations from a specified range.
- There is one **voltage monitor** for each channel which detects whether the supply voltage is leaving a specified range. Furthermore, both channels of the fail-safe system are protected against dangerous overvoltage.

Since the fail-safe system constitutes a supplement to a power converter, it is directly attached to this device via dedicated pins (not illustrated). These pins are used as power supply and for the communication with a DSP (Digital Signal Processor) that is part of the power converter and controls the electric motor. The fail-safe system and the DSP use a SPI (Serial Peripheral Interface) to communicate. This DSP is configured as SPI-master. The CPLDs are configured as SPI-slaves.

3.2.2 Fail-Safe System Behavior

When the system is in the safe state [GMSW10b], the safety-critical outputs are switched to ground (GND) and no power is applied to the motor. Consequently, the motor coasts and is not able to cause harm to people, environment or property. If the system is in the unsafe state, the safety-critical outputs are switched to V_{DD} and the motor can rotate. A rotating motor has the potential to cause harm. Figure 3.3 describes the behavior of the fail-safe system. In contrast to the unsafe state (MOTOR_RUNNING), the safe state can be divided into the states INITIALIZATION, PARAMETRIZATION, INIT_TEST, MOTOR_STOPPED and HARD_ERROR.

When the system is switched on, the CPLDs initialize their registers and flip-flops (INITIALIZATION). The behavior of the safety function SS1 depends on a parameter (SS1-time). Each CPLD receives a SS1-parameter from the DSP. The CPLDs store the SS1-time in volatile memory. Thus, this PARAMETRIZATION is necessary whenever the system is switched on.

When the parameterization is completed, the DSP notifies the CPLDs that a test (INIT_TEST) needs to be performed. This test verifies that the safety functions STO and SS1 are working properly.

If no fault was detected during the init-test, the DSP tells the CPLDs that the safe state can be quit (MOTOR_RUNNING).

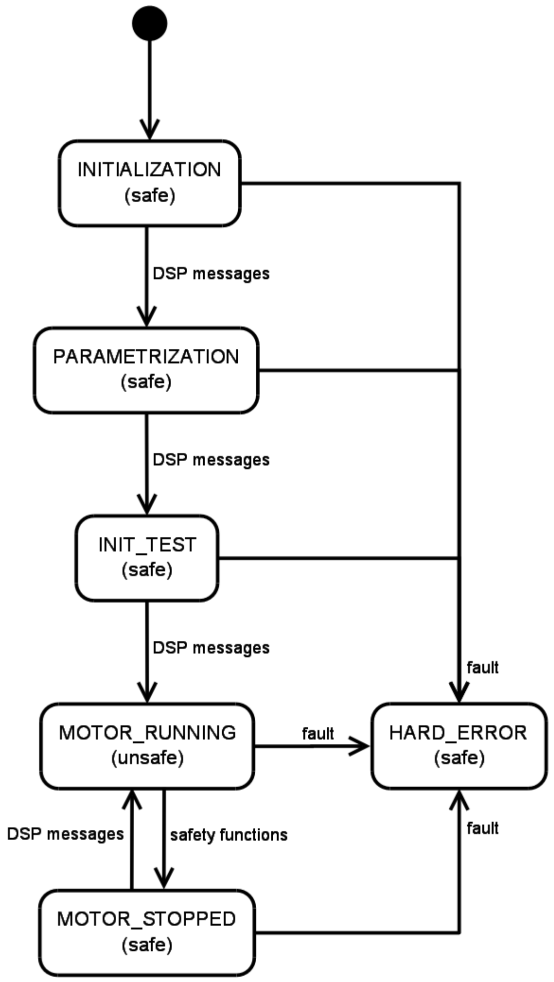


Figure 3.3: Behavior of the fail-safe system.

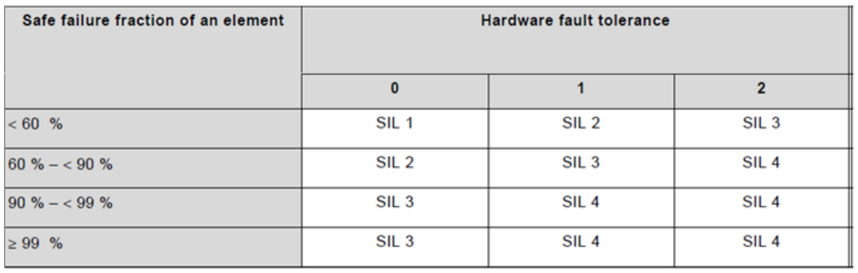
The safety functions STO and SS1 can be activated via the input stages. If a safety function is activated, the system enters the safe state after a specified amount of time (MOTOR_STOPPED).

If no fault occurred and the safety functions were performed properly and are not activated anymore, the DSP tells the CPLDs to quit the safe state. In this case, the motor can rotate again (MOTOR_RUNNING).

In all states, various safety integrity measures detect faults. If a fault is detected, the system enters the safe state (HARD_ERROR). In contrast to the other states, this state cannot be left until the system is switched off.

3.2.3 Safety Integrity Measures

IEC 61508 requires SIL 2 systems with a HFT of 1 to have a safe failure fraction (SFF) of at least 60%-90%. Also, EN ISO 13849 requires a medium diagnostic coverage (DC) for Cat 3, PL *d* safety functions. Thus, a number of measures to achieve sufficiently high SFF and DC have to be realized, which are performed concurrently by both channels of the fail-safe system.



Safe failure fraction of an element	Hardware fault tolerance		
	0	1	2
< 60 %	SIL 1	SIL 2	SIL 3
60 % – < 90 %	SIL 2	SIL 3	SIL 4
90 % – < 99 %	SIL 3	SIL 4	SIL 4
≥ 99 %	SIL 3	SIL 4	SIL 4

Figure 3.4: Connection between SFF and HFT, figure from [Com10].

The fail-safe system contains two channels which are able to execute the safety function STO and SS1 independently. The channels realize the implemented safety functions by using different means. This **diversity** reduces the probability of common cause failures.

The two CPLDs exchange corresponding pairs of signals. Some of these signal pairs contain information about the state of the safety functions. If a signal pair is discrepant for a certain time, a fault can be assumed. Thus, both CPLDs contain **discrepancy monitors** to signal a fault when the discrepancy of a signal pair exceeds a specified discrepancy time. Consequently, the system enters the safe state. The discrepancy monitors check for discrepancies as long as the fail-safe system is switched on.

The used type of CPLD is similar to an FPGA. It contains look-up tables and employs channel-based routing [LVHL04]. One of the advantages of CPLDs is their "instant-on" functionality. The used type of CPLD realizes this functionality using a configuration flash memory which is loaded into CRAM-cells on start-up of the device. The **init-test** is able to detect whether the safety functions can be performed properly. Thus, this test can detect faults in the flash memory, the configuration or the CRAMs, which affect the ability of a CPLD to perform the safety functions. This test is performed once after the parametrization of the CPLDs.

The periodic **SS1-test** starts to run as soon as the init-test is finished successfully. The SS1-test verifies the correct functionality and parametrization of the safety function SS1. Furthermore, the functionality of the counters for the safety function STO is tested. When a number of conditions are fulfilled (expected values of counters, signals and flip-flops), the CPLDs tell the DSP that the SS1-test needs to be quit. Consequently, the DSP tells the CPLDs to quit the SS1-test. If the DSP does not respond, the system enters the safe state. Thus, the CPLDs act as a watchdog for the DSP and vice versa.

The fail-safe system can use feedback signals to observe the state of the controlled motor. If the fail-safe system enters the safe state, the motor has to stop within a specified time. The **shut-down test** finds out whether this time is exceeded or not. If the specified time is exceeded, a fault is signaled. This test is detecting faults as long as the fail-safe system is switched on.

An undetected short circuit between the safety-critical outputs (first fault) is dangerous. If a second dangerous fault occurs (e.g. stuck-at-1 fault of a safety-critical output), none of the channels is able to correctly carry out the safety functions. Thus, we realize a **short circuit test** which is able to detect a short circuit between the safety-critical outputs. The test is activated whenever the system enters the unsafe state and is performed when the fail-safe system enters the safe state.

Each channel contains a **temperature monitor**. If a temperature monitor detects a temperature outside the specified temperature range, the system enters the safe state.

Each channel contains a **voltage monitor**. If a voltage monitor detects that the supply voltage is too high or too low, the system enters the safe state.

3.3 Safety Workflow

A V-model of the safety lifecycle of ASIC/FPGA/PLD designs is defined by the draft of IEC 61508-2. It requires a clearly structured development process. Figure 3.5 illustrates the V-model [GMSW10b] and the names of the tools which are used for the required phases.

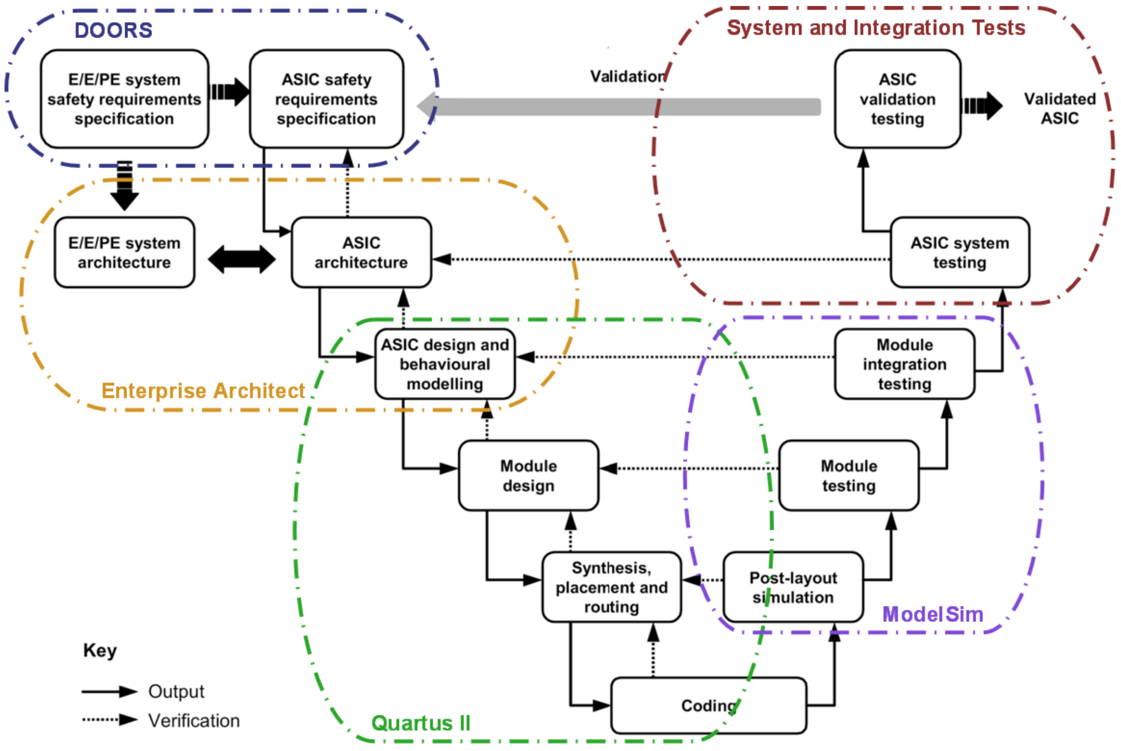

DOORS
E/E/PE system safety requirements specification
ASIC safety requirements specification
Validation
System and Integration Tests
ASIC validation testing
Validated ASIC
E/E/PE system architecture
ASIC architecture
ASIC system testing
ASIC design and behavioural modelling
Module integration testing
Enterprise Architect
Module design
Module testing
Synthesis, placement and routing
Post-layout simulation
ModelSim
Coding
Quartus II
Key
Output
Verification

Figure 3.5: V-model and tools

The result of each phase on the left-hand side of the V-model serves as input to the next phase on the left-hand side. In each phase, the results of the preceding phase are verified. Iterations between consecutive phases are possible. At first, the safety requirements for the E/E/PE (electric/electronic programmable electronic) system are specified and the requirements for the ASIC/FPGA/PLD are derived. For the development of the presented fail-safe system, the tool DOORS for requirements definition and management is used.

Later on, the architectures of the E/E/PE system and the ASIC/FPGA/PLD need to be defined. The tool Enterprise Architect in combination with a SysML [Hau06] extension to create a model of the entire E/E/PE system and the ASIC/FPGA/PLD is used. SysML [OMG10] diagrams are used to specify structure (Internal Block Diagram), relationships (Block Definition Diagram) and behavior (Use Case Diagram, Activity Diagram, State

Chart Diagram, Sequence Diagram). Moreover, the modules and the exact interfaces of the synthesizable VHDL-description, which defines the behavior of the CPLDs of the fail-safe system, are specified.

The definitions of the behavior and the modules follow. A synthesizable VHDL-description is created with the tool Quartus II . This tool is used to synthesize the VHDL-description and to perform place-and-route. After synthesis and place-and-route, a post-fit netlist is available. Quartus II [Alt10] is able to create post-fit netlist files which can be used for simulation tools and timing analysis tools.

Phases on the right-hand side of the V-model contain activities to verify the results of the phases on the left-hand side of the V-model. The post-fit netlist is verified using post-layout simulations. The draft of the IEC 61508-2 requires the alternative application of one of two equivalent techniques/measures to verify that the timing requirements for the circuit are fulfilled. A static timing analysis using a timing analysis tool is performed, which is part of Quartus II. According to the draft of the IEC 61508-2, it is also necessary to apply one of two equivalent techniques/measures to verify that no systematic faults occurred during synthesis and place-and-route. We perform simulations to verify the post-fit netlist against the synthesizable VHDL-description. The tool ModelSim is used for simulation.

Module testing is required to verify the correct implementation of the modules. At least one test bench is created for each module of the synthesizable VHDL-description. The test benches are executed using ModelSim. The draft of the IEC 61508-2 requires a test coverage greater than 99% for SIL 3. Furthermore, also ModelSim to evidence that statement coverage, branch coverage, condition coverage and expression coverage reach 100% for each module is used.

Module integration testing verifies that the implemented VHDL-description is correct. Again, test benches are created and ModelSim is used to simulate the entire VHDL-description. Some of the integration test cases are derived from use cases. Other test cases simulate the behavior of the entire VHDL-description in the presence of a fault.

After the integration of the components of the E/E/PE system, integration testing is performed to verify the ASIC/FPGA/PLD. During integration testing, also fault insertion tests are performed to verify the realized safety integrity measures. In order to perform fault insertion testing of the novel fail-safe system, it was necessary to develop a new method [GMSW09]. Finally system testing is performed to validate the entire system and the ASIC/FPGA/PLD.

3.4 Implementation

To define the behavior of the CPLDs, a synthesizable description is needed. From this description, a programming file can be created which is used to configure the CPLDs.

The draft of the IEC 61508-2 requires a restricted use of asynchronous constructs. Thus, the design is totally synchronous. Additionally, the draft of the IEC 61508-2 requires synthesizable descriptions to be highly modularized. Consequently, the synthesizable VHDL-description consists of various modules.

The number of required CPLD-resources is an important parameter as it directly determines the cost per fail-safe system. Hence, it is necessary to create a VHDL-description, which can be synthesized efficiently in terms of required CPLD-resources.

Due to the two independent channels of the fail-safe system, it is not necessary to implement resource-demanding concepts like safe input cells, safe output cells [AMF05] or totally self-checking circuits [BMSS00] to detect faults on the CPLDs. On the contrary, the CPLDs permanently test each other to detect faults. This guarantees sufficiently high safety integrity, while fewer CPLD-resources are required. Additionally, the synthesis tool is configured to synthesize circuits optimized for low area.

Due to optimizations, it is possible to implement the safety functions and the safety integrity measures using CPLDs, which contain only 240 logic elements (equivalent to 192 macrocells) per CPLD. The implementation requires approximately 95% of the available logic elements. The used CPLD [LVHL04] is the smallest member of its device family.

3.5 Fault-Insertion Testing

The IEC 61508 requires fault-insertion tests to be performed using the system hardware. This is why it is not sufficient to perform fault insertion tests using software simulations. Instead the system hardware (F-module) is used to perform the tests. The behavior of the CPLDs on the F-module is defined by a synthesizable VHDL-description. After the compilation of the VHDL-description, a programming file is available, which can be used to configure each CPLD via a JTAG interface. Usually, both CPLDs are configured using the same programming file. Nevertheless, it is possible to configure the CPLDs using different programming files. The presented method to perform fault-insertion testing exploits this fact. To inject a fault, we create a modified version of the synthesizable VHDL-description.

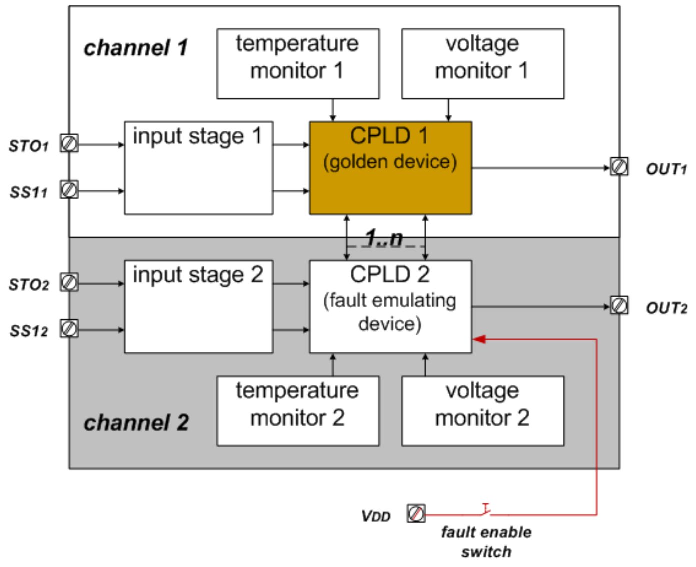


Figure 3.6: Fault-insertion Testing

This modified description makes it possible to inject a certain permanent or transient fault into one CPLD via an input pin of this CPLD. After this compilation, a modified programming file is available. One CPLD (fault-emulating device) is configured with this modified programming file. The other CPLD (golden device) is configured with the original programming file. Thus, it is possible to inject a fault into one CPLD but not into the other one. One input pin of the fault emulating device is connected to a manual fault-enable switch. When the input pin is not switched to VDD, no fault is injected and the fault emulating device has the same behavior as the golden device. If the input pin is switched to VDD, a fault is injected into the fault-emulating device.

To modify the VHDL-description, saboteurs can be added. This allows the insertion of transient faults like bit-flips. Also, permanent and intermittent faults like stuck-at faults, bridging faults and delay faults can be injected. Moreover, mutants can be added to the VHDL-description to inject faults. Saboteurs and mutants can be activated and deactivated via the fault-enable switch. Various fault models can be considered using the presented approach. The only restriction for the creation of saboteurs and mutants for this approach is the use of synthesizable VHDL-constructs.

Figure 3.6 illustrates the F-module, which is prepared for fault-insertion testing. In this case, CPLD 1 is configured as golden device and CPLD 2 is configured as fault-emulating device. Generally, it does not matter which CPLD is configured as golden device and which CPLD is configured as fault-emulating device.

3.5.1 Verification of Implemented Safety Integrity Measures.

According to the requirements of IEC 61508, the synthesizable VHDL-description is modular and consists of a number of interconnected components. We perform a system FMEA (failure modes and effects analysis) and a design FMEA to determine possible sources of failure of the components and their interconnections and identify the consequences in terms of system behavior.

If a fault of a component or interconnection leads to the safe state of the entire system, the fault can be considered to be safe. In this case, no fault-insertion test case needs to be planned. If a fault potentially leads to the unsafe sate of the entire system, we expect the implemented safety integrity measures to detect the fault and the system has to enter and/or maintain the safe sate. Consequently, a test case needs to be planned to verify that the safety integrity measures are able to detect the fault.

It is necessary to verify that each of the two channels is able to detect injected faults. Thus, CPLD 1 is configured as golden device for one half of the test cases, while CPLD 2 is configured as golden device for the other half of the test cases. For every test case, another fault has to be injected into the fault emulating device. Consequently another modified programming file needs to be generated and the fault emulating device needs to be reconfigured for every test case. Thus, a new modified synthesizable VHDL-description has to be created.

For every test case, we insert a saboteur between components or create a mutant of a component of the VHDL-description. This allows injecting a single fault into the fault-emulating device, while the golden device remains fault-free. Considered types of faults are stuck-at, bit-flip, stuck-then, stuck-else, assignment control and stuck-data. If the entire system (golden device in cooperation with the fault emulating device) detects the fault and consequently enters and/or maintains the safe state, the test is successful. Otherwise the test fails.

The synthesizable VHDL-description can be parameterized before compilation. The parameters are constants which are defined in a separate VHDL-package. There is one constant per fault-insertion test case. If all constants are set to 0, it is not possible to inject a fault into the CPLD after compilation and configuration via the fault-enable switch. The synthesized circuit contains no fault-insertion logic. Thus, in the VHDL description for the golden device, all constants are set to 0.

After the compilation and configuration, not a single logic element is required for mutants or saboteurs in this case. If the constant for a certain fault-insertion test case is set to 1, it is possible to inject a corresponding fault after the compilation and configuration using the fault-enable switch. In this case, the synthesized description of the fault-emulating device contains a saboteur or a mutant which can be activated via the fault-enable switch. Thus, for every fault-insertion test case, a single constant is set to 1 in the VHDL-description of the fault-emulating device. The remaining constants are set to 0. Consequently, there is only a little area overhead caused by a little fault-insertion logic required for a single saboteur or a single mutant for each test case.

There is no guarantee that the mapping of the modified VHDL-description on the logic elements of the CPLD is similar to the mapping of the unmodified VHDL-description on the logic elements. That is why the presented approach is limited to systems which contain at least two channels where each channel contains a CPLD or FPGA, which can be configured independently. There must always be at least one channel that contains a CPLD or FPGA that is configured with the compiled, unmodified VHDL-description.

For every test case, a certain fault is injected when the system is in one of two states. In the first case, a fault is injected when the F-module is in the state start-up and in the second case, a fault is injected when the F-module is in the state motor running. A test case is successful if the F-module either does not leave the state start-up (safe) or if it enters and maintains the safe state.

The following steps are necessary to verify all safety integrity measures:

1. Perform systematic FMEAs on system and design level to identify necessary test cases
2. Define expected results for each test case
3. Repeat for every test case
    (a) Insert a saboteur into the VHDL-description or create a mutant
    (b) Add a VHDL-constant to enable or disable the test case before compilation
4. Repeat for every test case
    (a) Edit VHDL-constants to enable insertion via the appropriate saboteur or mutant
    (b) Compile modified VHDL-description
    (c) Reconfigure the fault-emulating device
    (d) Inject fault when system is in the state start-up
    (e) Restart to inject fault in the state motor running
    (f) Decide if test was successful
    (g) Analyze and document result

The effort for all fault-insertion test cases is dominated by steps 1 and 2, which consume a lot of time for analysis and documentation. The time required for inserting saboteurs or mutants, editing VHDL-parameters, compiling the VHDL description and reconfiguring the fault emulating device is comparably low.

Chapter 4

Evaluations and Prototype

The applicability of the elaborated safety concept was evaluated using an industrial power drive system [GMSW10a]. This power drive system is able to control a DC motor. It converts a three-phase current to a direct current for low-power and high-power DC motors. The power drive system contains a microcontroller and a DSP (digital signal processor). These two devices realize standard functions that are able to control the DC motor in different manners. If the power drive system is not able to carry out these functions anymore due to a failure, a controlled motor cannot be stopped anymore and is able to harm people (e.g. worker in a factory). This is a risk. To reduce this risk and to make the power drive system applicable for even highly safety-critical applications, safety functions need to be realized accordingly to SIL 2 in adherence to IEC 61800-5-2 and IEC 61508. These safety functions reduce the risk of failing standard functions significantly. The safety functions are defined by IEC 61800-5-2 and were implemented using a CPLD-based fail-safe system (FSAR) based on the safety concept described by Chapter 3. The following safety functions, described in detail in the problem definition 1.2.3, are implemented:

- STO: Safe Torque-Off
- SS1: Safe Stop 1.

4.1 UC: Power Drive System

The application block diagram in Figure 4.1 depicts the main components of the PDS including the fail-safe system. The architecture of the power drive system is modular. There is a clear separation between the control electronic which is implemented in the Power Interface and the power electronic which is located at the Power Stage.

Moreover, the CPLD-based fail-safe system that is able to carry out the safety functions is an **optional** module. If no safety functions are required due to a low criticality of the application the power drive system is used for, it is able to work without the CPLD-based fail-safe system. In this case, the safety-relevant output signals STO_1 and STO_2 are permanently connected to V_{DD} and power can always be applied by the motor. If the power drive system is used for a highly critical application, the fail-safe system can be plugged into the power converter to ensure a high level of safety integrity. In this case, the

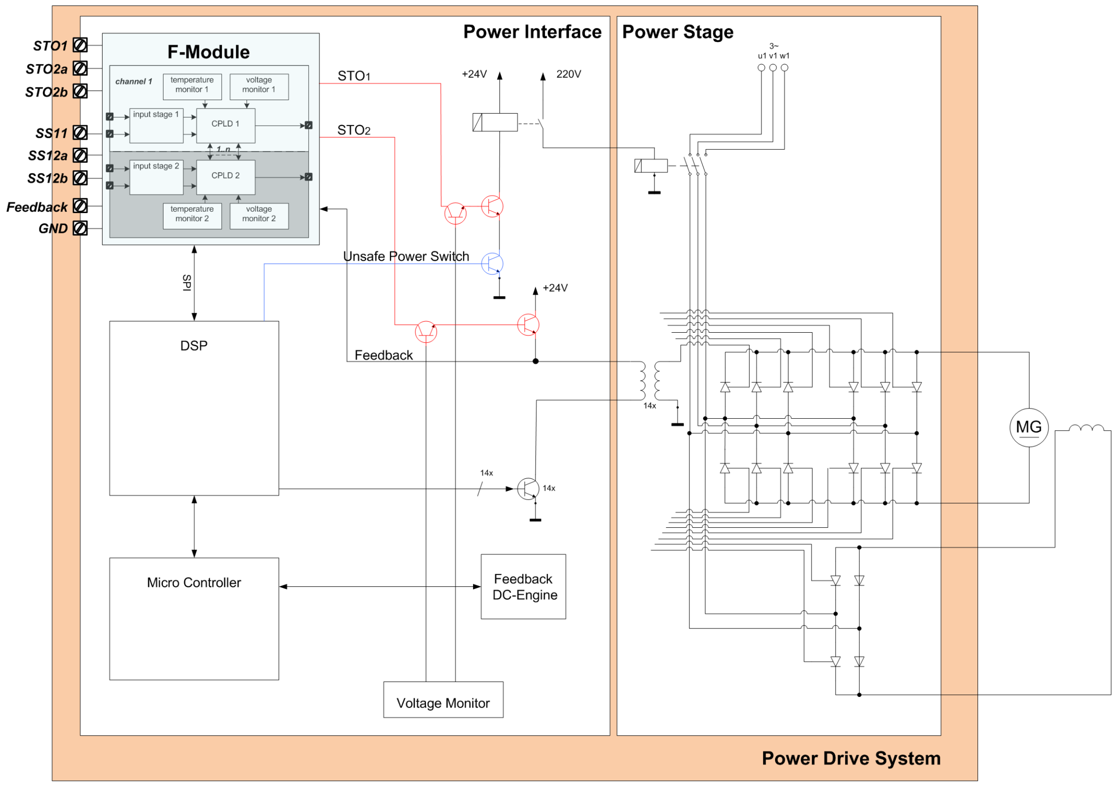


Figure 4.1: Power Drive System

safety-relevant output signals STO_1 and STO_2 are switched by the fail-safe system. The power drive system consists of the following components:

- Micro-controller: The micro-controller is the dominant component of the PDS system. It provides an interface to the user and to the higher-level infrastructure. Moreover, it continuously calculates the required voltage at the terminals of the DC motor depending on measured quantities of the DC engine and transmits the value of the required voltage to a DSP.
- DSP: The DSP is necessary to switch the thyristor bridges in real time to convert the three-phase voltage into the requested direct voltage.
- Voltage Monitor: A voltage monitor is necessary to protect the PDS system from destruction by overvoltage.
- Thyristor Bridge: The thyristor bridge is located at the power stage for voltage conversion.
- Transistor Switches: The transistor switches are controlled by the DSP to switch the thyristors. These switches are located at the power interface. They are electrically decoupled from the power stage.

- Feedback DC Engine: To reduce the calculation effort of the micro-controller, an ASIC is used to perform precalculations concerning measured electric fields, currents and voltages.
- F-Module: The F-Module is the fail-safe system that adopts the CPLD-based safety concept and realizes the safety functions accordingly to the applied standards.

4.2 F-Module

If plugged into the PDS, the F-Module communicates with the micro-controller via the DSP. Therefore, a SPI (serial peripheral interface) is established between DSP and F-module. The DSP alternatingly exchanges messages with the two CPLDs of the F-module. Whenever a CPLD receives a message from the DSP, it resets an internal counter. If this counter expires, a DSP failure is assumed and a safe state achieved. Hence, the CPLDs act as watchdogs for the DSP. Moreover, the SPI messages are used to inform the DSP and the micro-controller about the state of the F-module and to inform the F-module if a change of its state is requested (e.g. *from startup* to *power*).

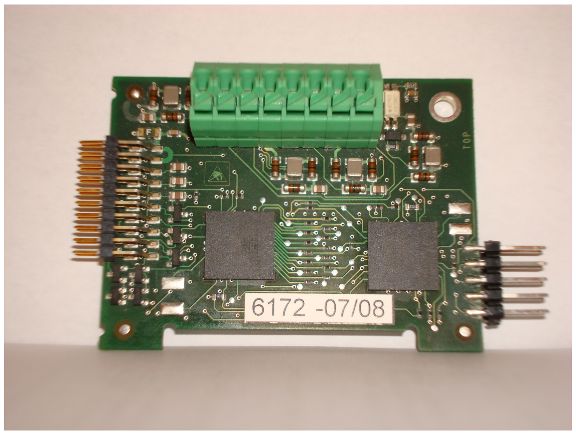


Figure 4.2: F-Module (see figure 2)

The SPI is also used for the parametrization of the safety function SS1 in state *startup* 3.3. In this state, the micro-controller transmits parameters via the DSP to each CPLD. The parameterization allows to vary the application-specific time delay of the safety function SS1 in *100ms* steps between *100ms* and *5min*. The possibility to parametrize SS1 is an important feature and it extends the range of possible applications significantly. Numerous applications originating from the high-power segment deal with very high currents. In these cases, a prompt turn-off of a machine without reducing the current before can damage expensive switches. Furthermore, an immediate turn-off can harm the production equipment or can destroy the products of the production process. Therefore, an application specific time delay is required to inform the micro-controller via the DSP about the demand of a safety function to give the PDS time to reduce currents and enable the high level infrastructure to ramp down the production process before a safe state is achieved.

The F-Module monitors the status of the two STO channels of the power drive system via feedback signals. The first feedback signal provides the status of the DC motor power switch and the second feedback signal provides the "enable" status of the thyristor bridge. If no "enable" status is available, no thyristor can be ignited and consequently, no power can be applied to the DC motor.

A hardware prototype of the F-Module is depicted in Figure 4.2. For the prototype of the F-module Altera MAX II, CPLDs were used [LVHL04]. The F-Module dedicated terminals (on top of the figure 4.2) to demand the safety functions with hardwired switches or via digital safety outputs of the higher-level infrastructure. The terminals on the left side of the figure are dedicated to power supply of the F-module and to communication with the DSP via SPI.

The design and the development of the F-Module were performed to meet the safety requirements and the requirements concerning price per piece. The key factor for a low unit price is the used CPLD with its number of available programmable logic elements. Thus, the design was developed to use as few logical elements as possible. Additionally, the synthesis tool was configured to synthesize circuits optimized for low area. Due to optimizations, it is possible to implement the safety functions and the safety integrity measures using CPLDs, which contain only 240 logic elements (equivalent to 192 macro cells) per CPLD. The implementation requires approximately 95% of the available logic elements. The used CPLD [Alt08] is the smallest member of its device family.

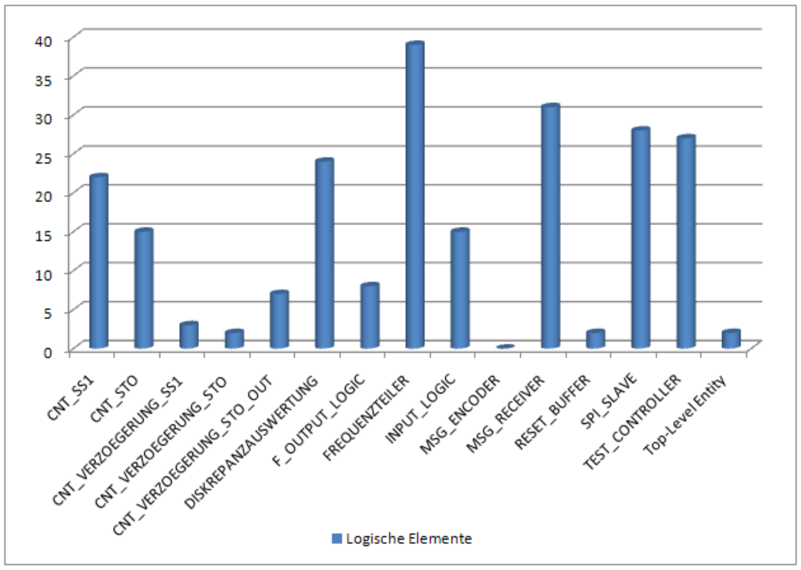


Figure 4.3: Distribution of used Logical Elements to the different functions. Figure from [Mad08].

Requirements of IEC 61508 second edition for the development process of safety-critical CPLDs were considered. Therefore, the prototype was developed according to a V-model using an appropriate tool chain. For the verification of the realized safety integrity measures, a new fault insertion testing technique was applied.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This work was motivated by the fact that industry demands cost-efficient concepts for the realization of safety functions for power drive systems in industrial automation. Therefore, a CPLD-based safety concept (FSAR) was elaborated that does not require the realization of microcontroller-specific software safety integrity measures like RAM tests or CPU tests that need a lot of effort in terms of design, implementation and verification as well as hardware resources (e.g. RAM, Flash, computing time).

Experiences and hints from other domains, technical reports as well as upcoming standards, regarding the design, implementation, tool chain and V&V [GM08], were considered to achieve an adequate level of safety integrity for the CPLD based fail-safe system.

The applicability of this safety concept for programmable logic devices (PLD, CPLD, FPGAs) was proved in terms of safety and cost-efficiency by its utilization for an industrial power drive application. The safety concept and its application were reviewed and assessed by an independent certification authority (TÜV SÜD). TÜV SÜD stated that the presented concept is suited to achieve **SIL 3** in adherence to IEC 61508 and IEC 61800-2 as well as **Cat 4, PL e** in adherence to EN ISO 13849. This means that the presented fail-safe architecture fulfills the requested SIL 2 requirements and is furthermore suitable for applications which request a higher safety integrity.

The method presented in 3.5 allows the application of a proven verification method (fault insertion testing) for a PLD-based fail-safe system. Methods to verify comparable micro-controller based systems using fault insertion testing are not applicable for the fail-safe system because the FSAR contains exclusively PLDs. It is possible to verify the safety-integrity measures of the fail-safe system using the system hardware (F-module). The successful verification of the realized safety integrity measures using fault insertion testing is an important prerequisite for the certification of the developed PLD-based fail-safe system in adherence to the desired standards. TÜV SÜD assessed a plan for validation and verification of the fail-safe system. This plan precisely describes the presented method to perform fault insertion testing. TÜV SÜD stated that the presented method is appropriate.

Finally, it needs to be underlined that the applicability of the presented "**Fail-Safe Architecture for Reconfigurable programmable logic devices**" (FSAR) is not limited to the automation domain. It can be concluded that the use of a PLD-based (PLD, CPLD or FPGA) safety concept is a competitive alternative to the use of micro-controller based safety concepts if comparably uncomplex safety functions need to be realized. In this case, the comparatively simple functionality does not justify a software implementation including the great effort for development of complex software safety integrity measures to make the use of micro-controllers acceptably safe.

5.2 Future Work

As stated in the introduction, the field of application and the complexity of embedded systems continuously increase. Future embedded systems have to consider novel requirements like cost efficiency, power awareness, intellectual properties, security, safety etc. In particular, with regard to the system, property safety of "future" embedded systems brings a lot of new challenges which need to be coped with in the future. These challenges can be grouped into (1) the application of new safety-related architectures and (2) the improvement of existing methods or the development of new methods to cover these new requirements.

5.2.1 Safety-Critical System Architectures

Depending on the field of application, we have to distinguish between two main streams of future safety-critical system architectures. The first main stream is the applicability of new technologies like dual-core or multi-core processors for safety-critical components, and the second main stream is the distribution of safety-critical functions to different components in safety-critical system.

New Technologies

Nowadays, there is a clear trend on the industrial sector towards dual-core or multi-core micro-processors with the aim of realizing two independent channels of a safety function in one micro-processor only. This will bring a significant reduction of costs and required space at printed circuit boards. In particular, the bottleneck of synchronization between the independent channels will be automatically solved with such a kind of architecture. However, due to the high integration of two or more cores into a single chip there is still the challenge to find adequate measures to control the effects on **Common Cause Failures**.

On the academic sector, the potential of different safety-related FPGA solutions are investigated. Thereby, the focus is set with multi-channel or multi-core implementations on a single FPGA. As mentioned in 2.2, methods and tools will be enhanced to support the safety argumentation and to get confidence in the applied tool chains. Furthermore, first analyses regarding safety integrity of single FPGA solutions were launched. However, as mentioned above, the problem of the **Common Cause Failures** is the limiting factor to apply such solutions in an industrial context.

Today, there are some promising micro-processor concepts available on the market. One concept is realized as lock step processor architecture. The lock step devices provide system-wide protection through seamless support for fault detection from the processor, through the bus interconnect, and into the memories. Such a concept significantly reduces (1) the implementation effort and (2) the required execution time for safety integrity functions (e.g. RAM tests). Furthermore, the producers of such micro-controllers claim that their micro-processors are suitable to achieve SIL 3 according to IEC 61508 or ASIL D according to ISO 26262. Despite to all these very promising properties of the lock step concept it has to be pinpointed that this concept has no real hardware fault tolerance. This disadvantage is expressed in the corresponding safety manuals of such processors where further measures are required, e.g. an additional external watchdog (e.g. Safety Manual TMS570LS20216S Device [Ins10]).

Another promising concept is the implementation of a “real” hardware fault tolerance in one micro-processor. Such activities were launched by SIEMENS. The basic idea is to implement two different cores in one device by taking the impact of common cause failures into account. Therefore, dedicated safety measures for power supply, error propagation etc. are required. Requirements to develop safety-critical systems based on ASICs, FPGAs or CPLDs are defined in the Second Edition of the IEC 61508 [Com10].

It can be summarized that – up to now – there are no micro-processors with a real hardware fault tolerance according to the safety standards available on the market. However, the currently available micro-processors facilitate the design of safety-critical systems which need to be applied in the industry.

Distributed Safety Architectures

Especially the automotive domain is very interested in developing “top-level” safety concepts for their applications. This is mentioned in the system architecture of today vehicles. A modern vehicle includes a lot of computing units which are connected via buses and wires. Due to the different suppliers of components a high diversity is inherently given, which reduces the impact of common cause failures. However this strategy has some challenges like available network bandwidth, fault response times, available controller throughput, synchronization issues etc. which have to be answered prior to an application of this approach. With the ARTEMIS R&D project Pollux [Pol10] a first initiative was started to cope with these challenges.

5.2.2 Safety Methods

In addition to "new" safety-critical system architectures, improved methods for the development of safety-critical systems are needed. The main topics which need to be addressed in the near future are (1) complexity issues, (2) component certification, (3) integrated and seamless tool chains and (4) the proper support of intellectual property rights in safety-related systems.

Complexity

The complexity is one of the greatest challenges in the future. New catch phrases like "System of Systems" or "Cyber-Physical Systems" express the need for interaction of different systems. Especially in the case of safety-critical systems where predictable system behaviors are required, new methodologies like system analysis are necessary to cope with arising complexity. However, one precondition to perform a system analysis is a description of the system model in a formal manner. Therefore, there are particular interests in development of methodologies which support on the one hand formal system verifications and on the other hand an applicability in an industrial context.

The European industry is aware of this necessity and already started different research projects to cover these topics. Examples are the ATTEST Projects [Con08] with their predecessor and successor projects to specify a domain-specific modeling language called EAST-ADL2 [ATE10, LF08, CFJ+08] or the CESAR Project [Con09] that aims at improving methods and tools for requirements engineering as well as component-based development.

Component Certification

Component-based certification and reusable certification are currently the topics that are highly promoted and discussed in the field of qualification and certification processes. The reason is that a significant part of the development resources are used for certification and qualification activities. In particular, reusability of certified components in combination with product line engineering promises a significant cost reduction in the development of safety-critical systems respectively safety-critical product families.

Tool Chains

As mentioned above, a huge development effort is used for certification and qualification activities. The reasons for that are the required consistency of all development artifacts over the entire product life-cycle, the traceability of requirements down to the implementation and the additional requirements to the V&V of the safety-critical system. Furthermore, a perceptible evidence which shows the fulfillment of all system and standard requirements, the so called *safety case*, is requested for a final certification or qualification. It is obvious that seamless tool chains can highly support the entire development of embedded system through the implicit consistence of development artifacts and the possibilities to perform impact analysis or automatically generated safety cases. However, the industrial applicability of holistic seamless tool chains is not available for the system-engineering domains. Today, mainly point-to-point integrations of major tools and proprietary solutions exist. A seamless tool chain is a clear requirement of future embedded system generations and it is one of the major challenges motivated by economical circumstances. One approach to overcome this problem is provided by the CESAR Project [Con09] which aims at defining a standard which supports the interoperability of tools for the application of seamless tool chains.

Intellectual Property Rights (IPR)

Due to the distribution of specialized know-how over different suppliers, an adequate protection of IP is requested. The AUTOSAR standard has already considered this need in its architecture. However, in the development of safety-critical systems methods and rules are needed to unambiguously define the behavior and safety properties of components without detail knowledge of their realization.

Chapter 6

Publications

This chapter contains the relevant publications and gives an overview as well as an assignment of contributions to the chaptors of this thesis, which explains the approach presented in Chapter 3 in detail.

Publication 1: *A Computer-Aided Approach to Preliminary Hazard Analysis for Automotive Embedded Systems*, ECBS - 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, Las Vegas, USA, 27–29 April 2011, [MGA^{+}11]

Publication 2: *Improving Automotive Embedded Systems Engineering at European Level*, E&I Elektrotechnik und Informationstechnik OVE-Verbandszeitschrift, Österreichischer Verband für Elektrotechnik, EU, June 2011, [GKA^{+}]

Publication 3: *WO2009080384A1- Method for actuating an DC machine*, AT, EU and US patent 2009, [GFH^{+}09]

Publication 4: *Design and Implementation of Safety Functions on a Novel CPLD-based Fail-Safe System Architecture*, ECBS - 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS'10), Oxford, UK, March 2010, [GMSW10b]

Publication 5: *CPLD basierende homogen redundante fehlersichere Architektur*, ME - Informationstagung Mikroelektronik, Vienna, Austria, October 2008, [GSW08]

Publication 6: *CESAR:Cost-efficient methods and processes for safety relevant embedded systems*, ARTEMIS - Embedded World , Nürnberg, Germany, March 2010, [GMP^{+}10]

Publication 7: *Model-based Toolchain for the Efficient Development of Safety-relevant Automotive Embedded Systems*, SAE - SAE World Congress, Detroid, USA, April 2011, [AGZ^{+}11]

Publication 8: *Fault Insertion Testing of a Novel CPLD-based Fail-Safe System*, DATE - Design, Automation and Test in Europe, Nice, France, April 2009, [GMSW09]

Publication 9: *A CPLD-based Safety Concept for Industrial Applications*, ISIE - Industrial Symposium on Industrial Electronics, Bari, Italy, June 2010, [GMSW10a]

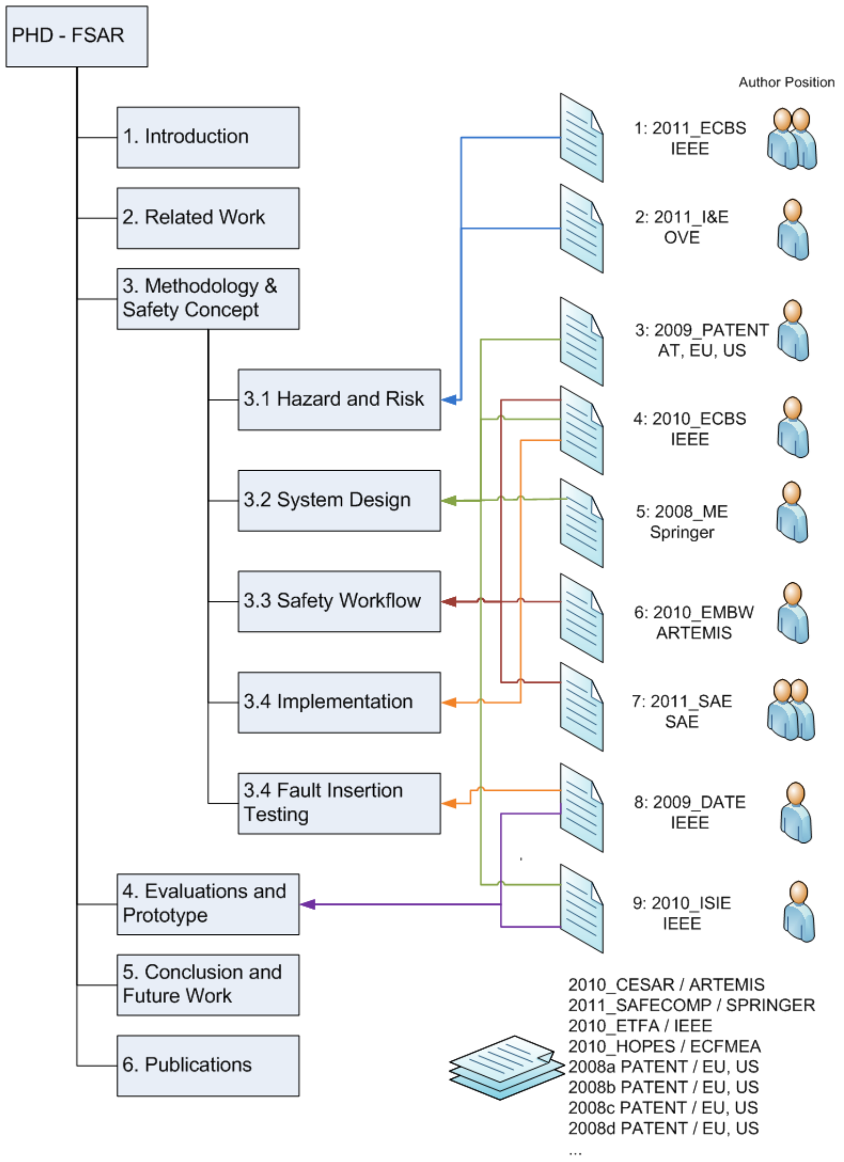


Figure 6.1: Assignment and contributions of publications (see figure 1).

2011 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems

A Computer-Aided Approach to Preliminary Hazard Analysis for Automotive Embedded Systems

Roland Mader[1,2], Gerhard Grießnig[1,2], Andrea Leitner[2], Christian Kreiner[2],
Quentin Bourrouilh[1], Eric Armengaud[3], Christian Steger[2], Reinhold Weiß[2]
[1]*AVL List GmbH, Austria*
[2]*Institute for Technical Informatics, Graz University of Technology, Austria*
[3]*Virtual Vehicle Competence Center, Austria*

***Abstract*—Powertrain electrification of automobiles leads to a higher number of sensors, actuators and control functions, which in turn increases the complexity of automotive embedded systems. The safety-criticality of the system requires the application of Preliminary Hazard Analysis early in the development process. This is a necessary first step for the development of an automotive embedded system that is acceptably safe. Goal of this activity is the identification and classification of hazards and the definition of top level safety requirements that are the basis for designing a safety-critical embedded system that is able to control or mitigate the identified hazards. A computer-aided framework to support Preliminary Hazard Analysis for automotive embedded systems is presented in this work. The contribution consists of (1) an enhancement for Preliminary Hazard Analysis to the domain-specific language EAST-ADL, as well as (2) the identification of properties that indicate the correct application of Preliminary Hazard Analysis using the language. These properties and an analysis model reflecting the results of the Preliminary Hazard Analysis are used for the automated detection of an erroneously applied Preliminary Hazard Analysis (property checker) and the automated suggestion and application of corrective measures (model corrector). The applicability of the approach is evaluated by the case study of hybrid electric vehicle development.**

***Keywords*-functional safety; ISO 26262; preliminary hazard analysis; safety goal; automotive embedded system**

I. Introduction

Nowadays cars contain embedded systems that incorporate up to 70 microcontrollers. These microcontrollers communicate via bus systems, gather sensor data or command actuators of the vehicle. At the same time the shift of the automotive industry towards powertrain electrification introduces new automotive sensors, actuators and functions. Electronic Control Units (ECUs) are responsible for managing the components (e.g. battery, motor) that can be found in HEVs (Hybrid Electric Vehicles) and components (e.g. engine, transmission) that can also be found in traditional vehicles. Therefore the safe operation of the vehicle depends on the correct operation of the embedded system.

Safety-critical automotive embedded systems are developed according to rigorous development processes. A necessary first step in such a development process is the application of *Preliminary Hazard Analysis* [1] (PHA) by a team of people with a wide variety of knowledge and skills. This analysis technique is applied earliest in the development process before neither concrete design solutions are elaborated nor enough numerical values are defined to allow the application of other techniques such as simulation or quantitative analyses. The purpose of PHA is the identification, classification and assessment of potential *hazards*[1] of a newly developed vehicle that are caused by potential failures of its embedded system. The early knowledge about the existence of hazards allows the definition of *safety goals* [2] (top-level safety requirements to the embedded system), even if detailed and quantitative information about the vehicle under development is insufficiently available. Considering the safety goals, a safety-critical embedded system design (typically including runtime fault detection capabilities) is developed that is able to best achieve the defined safety goals and to control or mitigate the identified hazards. Although the application of PHA alone cannot ensure safety of a vehicle, it is a necessary first step in order to eliminate or control hazards through adequate design, implementation, integration, verification and validation. Moreover the results of PHA serve as a baseline for later analyses. The early application of PHA is required by the safety standard ISO 26262 [3] for automotive E/E system development. This safety standard refers to this activity as *Hazard Analysis and Risk Assessment*.

Parallel to that, the field of model-based development (MBD) is rapidly evolving in order to manage product complexity, process complexity and organization complexity. Major application fields covered are communicating ideas and design, documenting and managing design information, automated model analysis and automated synthesis [4]. Potential benefits of these techniques are reduced time-to-market, reduction of costs and improved quality. Consequently, different approaches to early hazard analysis incorporate the use of *diagrammatic languages* [5] such as UML (Unified Modeling Language) in order to facilitate clear communication and documentation. However, existing approaches do not fully exploit the potential of diagrammatic languages for automated model analysis and automated syn-

[1]A hazard is defined as, "A state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event). [1]"

thesis. This hinders the early identification of imperfections that potentially affect the successful mitigation and control of hazards. Manual reviews and modifications of the analysis results are usually complicated and subject to mistake, and should be computer-aided.

The contribution of this work is a computer-aided approach to PHA in the development process of an automotive embedded system that is based on the domain-specific language EAST-ADL (Electronics Architecture and Software Technology-Architecture Description Language) [6]. Based on a detailed description of the work flow for PHA and defining safety goals, we propose (1) a language enhancement to EAST-ADL that allows to describe malfunctions and *operational situations* [2] in a more systematic way. Further, (2) we identify properties that indicate the correct application of PHA in the development process of a safety-critical automotive embedded system. We propose a tool implementation that can check these properties based on an analysis model reflecting the results of the PHA. This allows to automatically discover omissions that potentially affect safety of the vehicle under development. In the case a property is violated, the safety engineer automatically receives suggested solutions. Subsequently the most proper proposed solution can be accepted or it can be decided to solve the problem manually. If a suggested solution is accepted, the analysis model is automatically corrected.

This work is organized as follows: In Section II, related work is discussed. Section III describes the computer-aided approach to PHA including the work flow. In Section IV, the analysis model that is created in the course of PHA is described. In Section V and Section VI, our approaches to computer-aided checking and correction of the analysis model are described. Section VII outlines the experimental application of the approach using hybrid drive vehicle development. Finally Section VIII concludes this work.

II. Related Work

In the following, approaches to hazard analysis that are intended to be applied early in the development process are reviewed. The contributions [7], [8], [9], [10], [11], [12], [13] focus on defining systematic approaches that support the intellectual process of identifying and classifying hazards and defining means to mitigate or control them. All of them consider models to be a valuable aid for the application of hazard analysis techniques. None of these approaches refers to the use of diagrammatic languages to create models. In contrast we use a domain-specific, diagrammatic language for modeling to support coping with complexity, communicating ideas and facilitating automated model analysis and automated synthesis.

In [7] a technique called Actuator Based Hazard Analysis is proposed that can be carried out early in the development process, when only little information concerning the system implementation is available. The approach is based on the assumption that only the actuators of the system can affect their environment. The method defines three fault classes (commission, omission and stuck). Each system effect that describes an undesired enactment of an actuator is defined by a fault class, an analyzed actuator and an user intent. The method defines four severity classes (Catastrophic, Critical, Marginal and Negligible). All severity classes are applied to each actuator and the distribution between the severity classes is determined. Based on distribution and weighting, a criticality level can be determined that is the major input to the solvability analysis and design selection that allows to choose the design concept that is most likely to handle the identified hazards.

Another approach to preliminary hazard analysis for automotive systems that is similar to the one required by ISO 26262 [3] is described in [9]. The approach starts with hazard identification based on a system model. A PASSPORT diagram with supplementary descriptions is used as a system model. A further step of the approach is hazard classification according to severity, controllability and exposure.

In [10] an ISO 26262-compatible approach to preliminary hazard analysis is presented. The approach incorporates an architectural model and starts with (1) scope definition. In this phase, safety-critical functions of a vehicle are illustrated in a block diagram including control units, gateways, sensors, actuators and communication systems. The next step is (2) the definition of a role model. A control unit can contribute to multiple functions. In the context of different functions, the control unit may have different roles (e.g. actuation, calculation, monitoring). The next step is (3) the creation of a tabular architectural model. This starts with the mapping of functions to architectural elements. Subsequently, severity, exposure and controllability are evaluated and a resulting ASIL (Automotive Safety Integrity Level) is determined for each function. Then, roles (depending on functions) are assigned to each architectural element. Finally each architectural element has roles with corresponding automotive safety integrity levels.

The work proposed in [11] describes an approach to hazard analysis of safety-critical software-intensive systems called STPA (Systems Theoretic Process Analysis) for early application in the development process. The approach starts with the identification of hazards and related requirements or constraints. Subsequently inadequate control actions, control flaws, and inadequate control executions that lead to inadequate control actions are identified. This is input to a design process that aims on creating new constraints, refining existing constraints, creating a new design or modifying the existing design until all hazards are eliminated, mitigated or controlled. This process is iterative. The approach relies on a model that describes the control flow of the system under analysis and causes of accidents. The applicability of the approach is illustrated using a spaceflight application.

In [12] an approach to hazard analysis of SoS (System of Systems) is described that is intended to be applied early in the development process. This hazard analysis technique is focused on the interfaces between the particular systems. The approach is based on a model of the SoS as well as guidewords. In the course of the hazard analysis they carry out Input/Output Analysis as well as Network Analysis. The probability of the occurrence of accidents is assessed. A validation framework is established that incorporates the definition of goals. Based on these goals, metrics (e.g. percentage software safety requirements traceable to hazards) are defined that indicate the quality of the conducted hazard analysis. Some of the defined metrics depend on knowledge gained from previous analyses.

A new methodology to safety-critical system development is proposed in [13]. Amongst other activities, this methodology requires to identify those functions that are safety-critical. Thereafter hazards are identified, risks are assessed and risk mitigation means are defined and associated early in the development process. The work proposes metrics based on the identified hazards. An example is the metric *percentage software hazards* that is defined as number of software safety hazards divided by the number of system safety hazards. The approach is evaluated using a railway application.

In contrast to aforementioned related works, [14], [15], [16] explicitly refer to the use of diagrammatic languages to create models. In contrast to our approach, none of these approaches uses the annotated model to check for properties that indicate the correct application of the analysis technique.

An approach that combines hazard analysis and the use of a diagrammatic language to create models is described in [14]. The authors consider the use of UML models to be appropriate in order to handle the increasing complexity of safety-critical software systems. They use a subset of UML (component and deployment diagrams) to support hazard analysis at an early design stage. Boolean logic is used to formally model hazards and failure propagation. Starting with a component model of the system to be analyzed, (1) fault trees for all system hazards are derived. Subsequently (2) the propagation of component failures is analyzed for each component. Then (3) related behavior of deployment nodes and hardware devices has to be derived. Finally (4) boolean equations can be used to apply analysis techniques. The approach allows to identify the most serious hazards and failures and to determine components that require a more detailed safety analysis and assumed restrictions to fault propagation. This facilitates the systematic derivation of safety requirements.

The work described in [15] aims on improving the problems posed by the derivation of safety requirements and by conducting hazard analysis. The first step is the identification and description of functions associated with the level under study. Use cases and scenarios are used for function description. The second step is the failure identification. In this step a technique is applied that is inspired by techniques such as FHA (Functional Hazard Assessment) that is typically applied early in the development process and makes use of guidewords. In the third step, based on the analysis new safety related functional requirements are identified. The approach was evaluated using an use case from the avionics domain.

An approach to preliminary hazard analysis using EAST-ADL is proposed in [16]. The proposed workflow starts with the description of the functions (e.g. Cruise Control) of the vehicle, their operation needs and other stakeholder requirements. Thereafter a feature tree model is used to structure the vehicle functions. After the allocation of requirements to the features, the vehicle is well determined in terms of its requirements, functions and modes. This is the input to the identification and classification of hazards based on the functions and their related requirements. Thereafter safety goals are derived that constitute top level safety requirements.

In contrast to aforementioned related works, approaches that allow to conduct hazard analysis early in the development process in the context of more sophisticated MBD tool support are defined in [17], [18], [19]. These approaches provide a framework to support modeling using a domain-specific, diagrammatic language as well as the definition and automated checking of properties, but none of them defines properties that support the conduction of hazard analysis. In contrast our approach supports computer-aided checking, based on well defined properties that are presented in this work and allows the automated correction of the analysis model.

The work proposed in [16] in combination with the tools [17] and [18] allows the definition of properties using OCL (Object Constraint Language). The combined approach allows the definition and checking of properties on demand.

Tools that aim to support the safety standard ISO 26262 are reviewed in [19]. Among the reviewed tools is a tool named Medini Analyze following an MBD approach. It supports the definition of vehicle functions and the application of hazard analysis early in the development process. The tool allows to define constraints using the OCL language that can be automatically validated on demand. Besides a predefined set of checking rules, users can define their own rules.

III. Computer-Aided Preliminary Hazard Analysis

The application of PHA early in the development process of an automotive embedded system when less detailed and quantitative information about the vehicle under development is available is a cornerstone in the development process of an embedded system that is acceptably safe. The identified and classified hazards and the derived safety goals determine

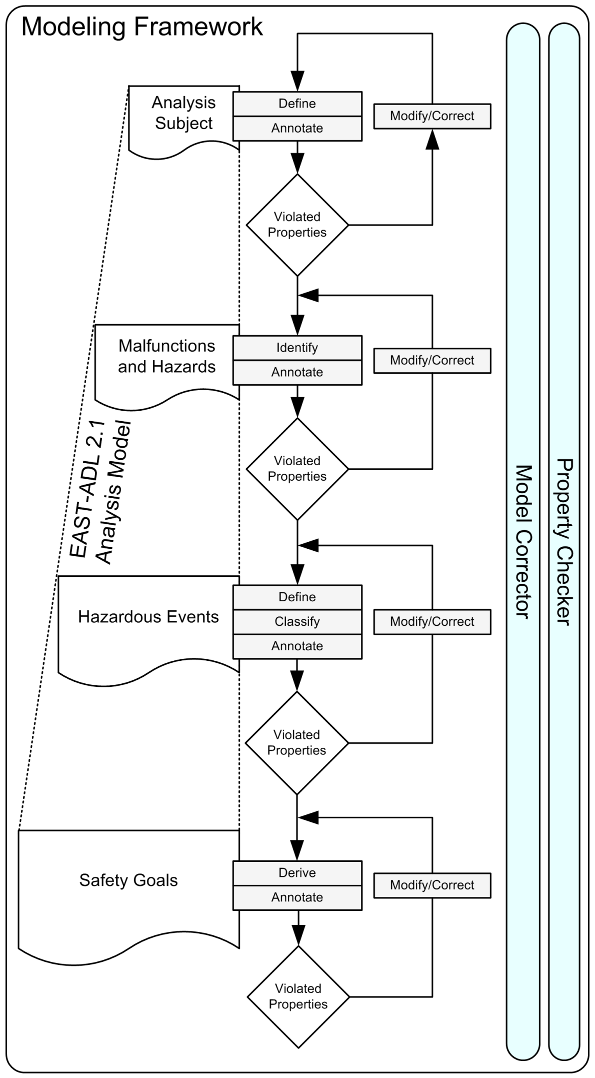


Figure 1. Computer-Aided Approach to Preliminary Hazard Analysis.

the design of the safety-critical embedded system including its fault detection capabilities.

The proposed methodology for PHA is based on the work described in [16] and is depicted in Figure 1. In this approach an analysis model is annotated, systematically enhanced and refined using the domain-specific language EAST-ADL until the results of the PHA are satisfactory. This analysis model describes the vehicle under development, the results of the PHA as well as the derived safety goals. We use EAST-ADL including an enhancement (see also Section IV-A and Section IV-B). The proposed methodology is explained in the following:

1) **Definition of the Analysis Subject:** First information concerning the vehicle under development is collected and modeled. Functions of the vehicle (e.g. motoring or recuperative braking) are described. In addition, requirements are associated with these functions (e.g. requirements for activation or deactivation). In addition modes (e.g. drive, creep or acceleration) are associated with each function and with relevant requirements.
2) **Identification of Malfunctions and Hazards:** Based on the definition of the analysis subject, possible malfunctions of the vehicle are identified. This is carried out using guidewords (see Section IV-A). Each function is analyzed with regard to each guideword. When a guideword applies to a function, the violated requirements are identified. Thereafter the malfunction (e.g. *unintended* positive torque) is described and associated with the analysis subject and the violated requirements. Hazards (e.g. unintended acceleration of the vehicle) are derived for each malfunction and associated.
3) **Definition and Classification of Hazardous Events:** The next step is to identify typical traffic situations (oncoming traffic on a highway in a curve), maintenance situations (e.g. vehicle at lifting ramp) (see Section IV-B) as well as other operational situations. Moreover use cases that describe the behavior (e.g. overtaking or changing oil) of the related actors (e.g. driver or mechanic) are described. Thereafter *hazardous events* [2] are determined as relevant combinations of hazards, use cases and operational situations. Subsequently relevant modes are identified for each hazardous event and associated with it. The criticality of each hazardous event is assessed in terms of its *controllability*, *severity* and *exposure* [2] and an *ASIL* [3] (Automotive Safety Integrity Level) is assigned. This is underpinned by the definition of classification assumptions.
4) **Derivation of Safety Goals:** For each hazardous that has a sufficiently high ASIL, a safety goal is defined and associated. A safe state is defined (e.g. switch open) for each safety goal. Alternatively a safe mode (e.g. limp home mode) is determined for each hazardous event. These safety goals are the top level safety requirements to the safety-critical automotive embedded system.

After the completion of these working steps lower-level safety requirements can be derived from the safety goals and an embedded system architecture including fault detection capabilities can be defined. Based on these requirements, software and hardware of the safety-critical embedded system are implemented, integrated, verified and validated. However, these steps of the development process of a safety-critical automotive embedded system are beyond the scope of this paper.

The application of PHA for contemporary vehicles is challenging. The analysis subject is complex and potentially contains many sensors, actuators and functions, whose combination can lead to a multitude of hazards caused by failures of the embedded system. This complexity results in a large set of information to manage during PHA, and the analysis model grows in size. The exhaustive application of PHA and the projection of its results onto an analysis model that is complete, consistent and allows traceability is cumbersome and error prone.

Challenges are (1) the thorough understanding of the system functionalities and environment, (2) the correct application of a method for the systematic identification and classification of the related hazards including the derivation of safety goals, and (3) the correct modeling using the domain-specific language.

Therefore we propose to aid the process of applying PHA and creating the analysis model by automated property checking (Property Checker) as well as automated correction (Model Corrector) in order to detect an erroneously applied PHA and to enable corrections (see Section V and Section VI). Although not able to conduct PHA automatically, this approach strongly supports the process of PHA in providing a guidance for its application and the reflection of its results on the analysis model. This guidance allows identifying omissions, inconsistencies and missing traceability links during PHA and supports the creation of a consistent and complete set of safety goals.

IV. Analysis Model

In the course of the PHA, results are annotated using the language EAST-ADL (see Figure 1). This language is domain-specific and tailored to the needs of automotive embedded systems development. It is diagrammatic such as UML and consist of syntactic elements such as boxes, ovals, lines or arrows. Its *abstract syntax* is defined by its meta model and its *semantic domain* and *semantic mapping* [5] are defined using natural language. The EAST-ADL specification can be found in [6].

EAST-ADL allows to describe concepts relevant to the application of PHA in accordance with the safety standard ISO 26262. It allows the definition of a vehicle in terms of its functions (meta class VehicleFeature), modes (meta class Mode) and requirements (meta class Requirement). By using these modeling concepts, the analysis subject can be described. Furthermore meta classes are available that allow to describe malfunctions (meta class FeatureFlaw) and resulting hazards (Hazard). Moreover operational situations (meta class OperationalSituation) can be defined. The behavior of the relevant actors in the context of the operational situation can be described with use cases. A combination of hazard, mode, operational situation and use case defines a hazardous event (meta class HazardousEvent). Moreover top level safety requirements (meta class SafetyGoal) can be defined for hazardous events. Associations between the aforementioned concepts can be created to precisely define their relations and assure proper traceability.

We propose an enhancement to EAST-ADL that allows to define malfunctions and operational situations in more detail. The enhancement is defined in a similar manner as EAST-ADL in the following. The relation of the enhancement's meta model to EAST-ADL is depicted by Figure 2.

A. Language enhancement: Malfunctions

The use of guidewords has been proved to be beneficial and is also applied by approaches such as [7], [12], [15]. In our approach, the aim of the use of guidewords is to systematically identify potential malfunctions of the vehicle.

EAST-ADL does not include predefined guidewords to classify malfunctions (meta class FeatureFlaw). Therefore EAST-ADL's meta class FeatureFlaw has been extended to support the use of guidewords. Using our enhancement, malfunctions (meta class GuidedFeatureFlaw) can be characterized by the following guidewords.

- **No:** The vehicle omits to carry out a function although it is demanded.
- **Unintended:** A function is carried out without demand.
- **Reverse:** The vehicle carries out a function but fails in applying it in the demanded direction.
- **More:** The vehicle carries out an intended function but exceeds the demanded degree of intensity.
- **Less:** The vehicle carries out an intended function but falls below the demanded degree of intensity.
- **Other:** If none of the guidewords above are able to characterize the malfunction.

B. Language enhancement: Operational Situations

An operational situation is characterized by conditions external to the vehicle. If a hazard inevitably leads to an accident depends on the operational situation in context as well as on the behavior of the involved actors. As an example, if the vehicle brakes without demand of the driver (hazard) while the vehicle waits in front of a crosswalk and a pedestrian crosses, the result is harmless. In contrast if the same hazard occurs while the vehicle moves at high speed on a curvy road, the hazardous event can lead to a lethal accident. From this example it becomes obvious that an acceptably precise definition of operational situations is a cornerstone for the estimation of severity, exposure and controllability of the corresponding hazardous events since they determine the ASILs of the derived safety goals. The necessity of precisely describing operational situations in the context of hazards is also emphasized by other approaches [11], [15].

EAST-ADL rudimentarily supports the definition of generic operational situations. Frequently occurring operational situations are traffic situations (the vehicle is driven) and maintenance situations (the vehicle is being repaired or serviced). Therefore we propose an enhancement to EAST-ADL to facilitate a more accurate definition of traffic situations and maintenance situations. The meta classes TrafficSituation and MaintenanceSituation are derived from EAST-ADL's OperationalSituation. Each of these meta classes

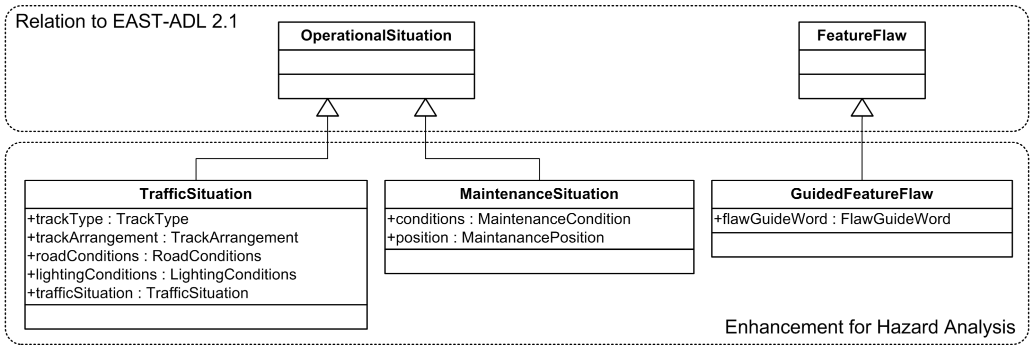

Relation to EAST-ADL 2.1
OperationalSituation
FeatureFlaw
TrafficSituation
+trackType : TrackType
+trackArrangement : TrackArrangement
+roadConditions : RoadConditions
+lightingConditions : LightingConditions
+trafficSituation : TrafficSituation
MaintenanceSituation
+conditions : MaintenanceCondition
+position : MaintanancePosition
GuidedFeatureFlaw
+flawGuideWord : FlawGuideWord
Enhancement for Hazard Analysis

Figure 2. Proposed enhancement to EAST-ADL for Preliminary Hazard Analysis (referenced enumerations are textually described).

can be described using attributes. For each attribute an enumeration is defined. Traffic situations can be coarsely characterized by following attributes:

- **Type of track:** The kind of the underlying track the vehicle is operating on such as highway, road, city street, mountainous track, crossings, drive-up, descent or parking area.
- **Track arrangement:** The geometry of the track the vehicle is operating on such as curve or straight.
- **Road conditions:** The road conditions the moving vehicle is exposed to because of the weather such as icy, normal, snowy or wet.
- **Lighting conditions:** The conditions that depend on the time of the day and the weather such as good visibility, medium visibility or poor visibility.
- **Traffic situation:** The conditions caused by other road users such as stop and go, crashing vehicles, colliding vehicles, pedestrian crossing, vehicle crossing, vehicle overtaking, adjacent vehicle, congestion, platoon traffic, oncoming traffic, no other road users, game crossing or street workers present.

Maintenance situations (MaintainanceSituation) can be coarsely defined by following attributes.

- **Maintenance position:** The position of the vehicle during maintenance such as lifting ramp or ground.
- **Maintenance conditions:** The conditions the maintenance operation is exposed to such as no other people or other people around.

Besides the defined value range, each enumeration contains the value *undefined*. This allows to express that a certain attribute is irrelevant in context of a hazardous event. This helps to collapse the number of traffic situations and maintenance situations to be dealt with during PHA.

V. Automated Property Checking

The analysis model that is annotated using EAST-ADL including the proposed enhancement reflects the results of the PHA. We propose properties that *indicate* the correct application of PHA, if they are fulfilled by the analysis model. If these properties are violated, the correct application of PHA is *not assured*. Note that the fulfillment of all the properties does not guarantee the exhaustiveness of the PHA, since the correct and complete understanding of the system by the team of people that applies PHA cannot be verified with this approach. This approach can show the erroneous application of PHA, but it cannot prove the absence of errors.

The defined properties are automatically checked using the evolving analysis model (see Figure 3). A property checker uses the definition of properties to continuously check the analysis model while PHA is carried out. It automatically identifies modeling elements that violate properties and presents a list of violating modeling elements and the properties affected. This information allows to early identify errors in the PHA before they can affect the definition of an adequate set of top level safety requirements to the safety-critical embedded system. Since automated checking is performed concurrently with the application of PHA and the creation of the analysis model, the property checker is also a valuable guide while carrying out PHA.

Input to the definition of these properties was (1) the domain-specific language EAST-ADL including the enhancements defined above and (2) the automotive safety standard ISO 26262 [3]. This standard defines requirements to the application of PHA for vehicles. Moreover the standard defines how the criticality of hazardous events shall be classified in terms of severity, exposure and controllability and defines how the required safety integrity level shall be derived. Moreover it is defined how safety goals shall be derived from defined hazardous events.

The properties are listed in Table I. Column *Meta Class* denotes the meta class of the enhanced EAST-ADL language (see Section IV) that can violate the corresponding property. Column *Property Definition* defines the properties for the corresponding meta classes in natural language.

Assume M is an analysis model, M_{MM} is the meta model of the enhanced EAST-ADL language and P is the set of properties as defined in Table I. Assume e is a modeling element of the analysis model, t is a type defined by the meta model of the analysis model and p is a property (Expression 1).

Property ID	Meta Class	Property Definition
0	Item	A complementary description has been defined
0a	Item	At least one VehicleFeature has been defined
1	Item	At least one Hazard has been identified
2	Item	At least one FeatureFlaw has been identified
0b	VehicleFeature	A complementary description has been defined
0c	VehicleFeature	Associated with at least one Item
23	VehicleFeature	A Requirement is satisfied
21	Requirement	An ID is defined
22	Requirement	A requirements text is defined
24	Requirement	Is satisfied
26	Mode	A condition has been defined
3	FeatureFlaw	At least one Hazard is identified
4	FeatureFlaw	Associated with at least one Item
5	FeatureFlaw	A complementary description has been defined
6	Hazard	At least one FeatureFlaw is associated
7	Hazard	At least one Item is associated
8	Hazard	At least one HazardousEvent has been identified
9	Hazard	A complementary description has been defined
10	HazardousEvent	At least one Hazard is associated
11	HazardousEvent	At least one UseCase is associated
12a	HazardousEvent	At least one SafetyGoal is associated if ASIL greater than QM
13	HazardousEvent	Associated with at least one OperationalSituation
14	HazardousEvent	ASIL has been correctly derived from Controllability, Severity and Exposure
17	HazardousEvent	Classification assumptions have been defined
25	HazardousEvent	Associated with at least one Mode
15	SafetyGoal	A HazardousEvent is associated
16	SafetyGoal	A safe state is defined
18	SafetyGoal	The ASIL has been correctly derived from associated HazardousEvents
20	OperationalSituation	A complementary description has been defined
27	All	A name must be assigned

Table I
PROPERTIES OF THE ANALYSIS MODEL THAT ARE AUTOMATICALLY CHECKED WHILE PRELIMINARY HAZARD ANALYSIS IS APPLIED.

$$e \epsilon M, t \epsilon M_{MM}, p \epsilon P \tag{1}$$

Moreover, $I(e,t)$ pertains, if e is of type t, $D(t,p)$ pertains, if p is defined for type t and $H(e,p)$ pertains if property p holds for modeling element e. If a model M *indicates* the correct application of PHA, Expression 2 is valid. In this case, no modeling element violates a property.

$$\neg\exists e \neg\exists t \neg\exists p (I(e,t) \wedge D(t,p) \wedge \neg H(e,p)) \tag{2}$$

If a model M *shows the erroneous application* of PHA, Expression 3 is valid. In this case, at least one modeling element violates a property.

$$\exists e \exists t \exists p (I(e,t) \wedge D(t,p) \wedge \neg H(e,p)) \tag{3}$$

VI. COMPUTER-AIDED MODEL CORRECTION

If violated properties indicate an erroneous application of PHA, corrective measures need to be carried out. To ease the correction of errors, we propose to support the identification of a proper correction measure. This can be achieved by consulting the model corrector that suggests possible solutions depending on the violated property and the current analysis model. Input to the definition of suggestions are again ISO 26262 and the enhanced language EAST-ADL. Depending on concerned model elements and violated properties, the model corrector identifies possible solutions (see Figure 3). If the user decides to accept a solution, the model is automatically modified accordingly.

Suggestions depending on the violated property are listed in Table II. Column *Meta Class* denotes the meta class of the enhanced EAST-ADL language that can be subject to the suggestion of an automated correction. Column *Suggestion* defines the possible suggestions for the corresponding meta classes in natural language.

Assume M is an analysis model, M_{MM} is the meta model of the enhanced EAST-ADL language, P is the set of properties as defined in Table I and S is the set of suggestions as defined in Table II. Assume e_1 is a modeling element of the analysis model, t_1 is a type defined by the meta model, p_1 is a property and s_1 is a suggestion (Expression 4).

$$e_1 \epsilon M, t_1 \epsilon M_{MM}, p_1 \epsilon P, s_1 \epsilon S \tag{4}$$

Assume that before an automated model correction is carried out (precondition), e_1 is of type t_1 and violates p_1 that is defined for type t_1 (Expression 5).

$$I(e_1,t_1) \wedge D(t_1,p_1) \wedge \neg H(e_1,p_1) \tag{5}$$

If the user accepts suggestion s_1, the analysis model M is

Property ID	Meta Class	Suggested Solution
0	Item	Creation and association of Comment
0a	Item	Creation and association of VehicleFeature
0a	Item	Associate one of the VehicleFeatures without Item
1	Item	Creation and association of Hazard
1	Item	Associate one of the Hazards without Item
2	Item	Creation and association of FeatureFlaw
2	Item	Associate one of the FeatureFlaws without Item
0b	VehicleFeature	Creation and association of Comment
0c	VehicleFeature	Creation and association of Item
0c	VehicleFeature	Associate one of the Items
23	VehicleFeature	Creation and association of Requirement
23	VehicleFeature	Associate one of the unsatisfied Requirements
24	Requirement	Creation and association of VehicleFeature
24	Requirement	Associate one of the VehicleFeatures without Requirement
3	FeatureFlaw	Creation and association of Hazard
3	FeatureFlaw	Associate one of the Hazards without FeatureFlaw
3	FeatureFlaw	Associate one of the Hazards with FeatureFlaw
4	FeatureFlaw	Creation and association of Item
4	FeatureFlaw	Associate one of the Items
5	FeatureFlaw	Creation and association of Comment
6	Hazard	Creation and association of FeatureFlaw
6	Hazard	Associate one of the FeatureFlaws without Hazard
6	Hazard	Associate one of the FeatureFlaws with Hazard
7	Hazard	Creation and association of Item
7	Hazard	Associate one of the Items
8	Hazard	Creation and association of HazardousEvent
8	Hazard	Associate one of the HazardousEvents without Hazard
9	Hazard	Creation and association of Comment
10	HazardousEvent	Creation and association of Hazard
10	HazardousEvent	Associate one of the Hazards without HazardousEvent
10	HazardousEvent	Associate one of the Hazards with HazardousEvent
11	HazardousEvent	Creation and association of UseCase
11	HazardousEvent	Associate one of the UseCases
12a	HazardousEvent	Creation and association of SafetyGoal
12a	HazardousEvent	Associate one of the SafetyGoals without HazardousEvent
13	HazardousEvent	Creation and association of OperationalSituation
13	HazardousEvent	Associate one of the OperationalSituations
14	HazardousEvent	Correction of the ASIL according to the requirements of ISO 26262
25	HazardousEvent	Creation and association of Mode
26	HazardousEvent	Associate one of the Modes without HazardousEvent
15	SafetyGoal	Creation and association of HazardousEvent
15	SafetyGoal	Associate one of the HazardousEvents with ASIL larger than QM and without SafetyGoal
18	SafetyGoal	Modification of the ASIL according to the ASIL of the corresponding HazardousEvent
20	OperationalSituation	Creation and association of Comment

Table II
POSSIBLE SOLUTIONS TO PROBLEMS THAT ARE AUTOMATICALLY SUGGESTED DEPENDING ON THE ANALYSIS MODEL.

automatically corrected and transformed to analysis model M' by function γ depending on M, e_1, t_1, p_1 and s_1 (Expression 6).

$$\gamma(M, e_1, t_1, p_1, s_1) \rightarrow M' \qquad (6)$$

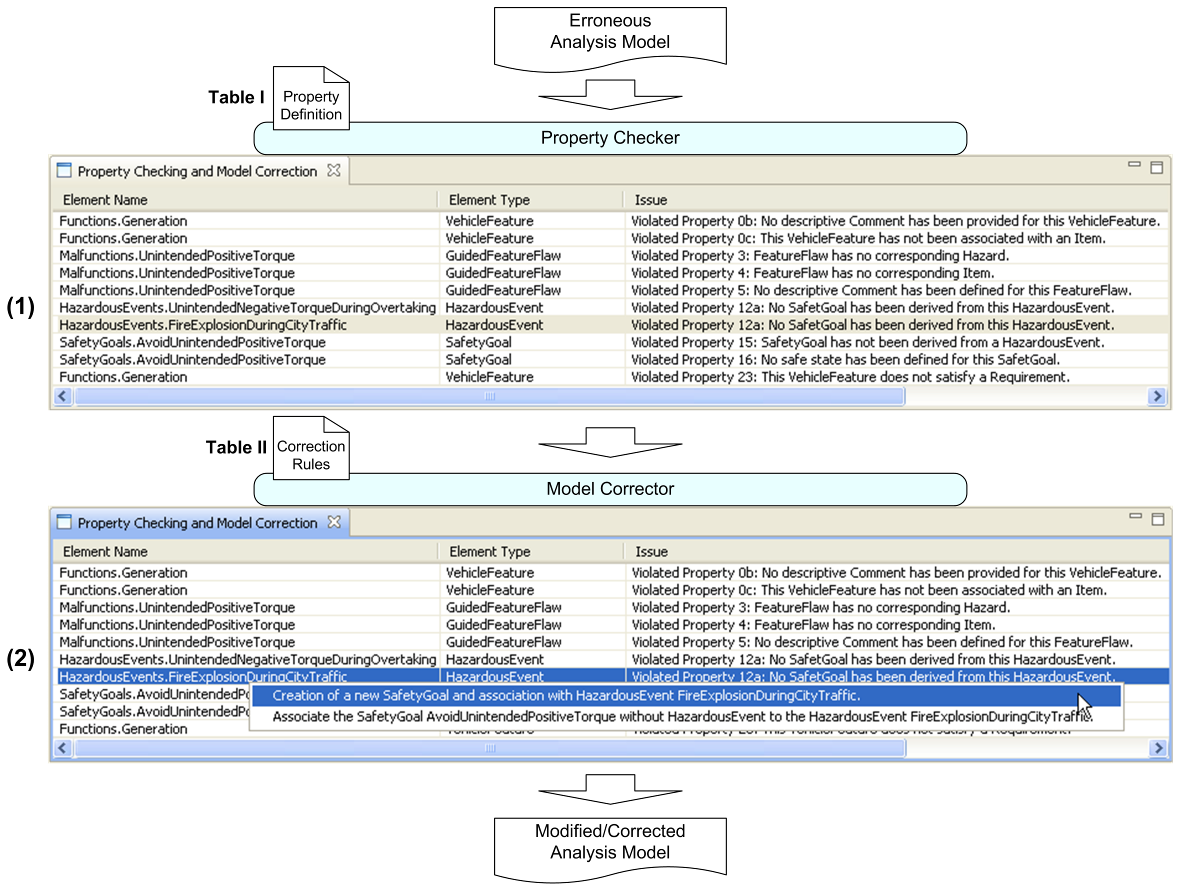
After the modification (postcondition) e_1 is an element of M', e_1 is still of type t_1 and does not violate p_1 any more (Expression 7).

$$e_1 \epsilon M', I(e_1, t_1) \wedge D(t_1, p_1) \wedge H(e_1, p_1) \qquad (7)$$

VII. EXPERIMENTAL EVALUATION

An Eclipse-based open source tool named Papyrus [17] is available that facilitates UML-modeling as well as the definition of UML profiles. An open source plugin is available for Papyrus [18] that allows the creation of EAST-ADL models. This plugin was enhanced to support the creation of analysis models such as described in Section IV. Another plugin for the Papyrus tool was developed that facilitates property checking as well as model correction such as proposed in Section V and Section VI.

Thereafter the proposed approach to PHA of automotive embedded systems was experimentally evaluated by the case study of HEV [20] development. One of the main characteristics of this type of vehicle is the addition of an electric motor that supports the classic combustion engine providing supplementary or substitutive positive torque. If such a vehicle uses its E-motor to support the combustion engine, it



Element Name	Element Type	Issue
Functions.Generation	VehicleFeature	Violated Property 0b: No descriptive Comment has been provided for this VehicleFeature.
Functions.Generation	VehicleFeature	Violated Property 0c: This VehicleFeature has not been associated with an Item.
Malfunctions.UnintendedPositiveTorque	GuidedFeatureFlaw	Violated Property 3: FeatureFlaw has no corresponding Hazard.
Malfunctions.UnintendedPositiveTorque	GuidedFeatureFlaw	Violated Property 4: FeatureFlaw has no corresponding Item.
Malfunctions.UnintendedPositiveTorque	GuidedFeatureFlaw	Violated Property 5: No descriptive Comment has been defined for this FeatureFlaw.
HazardousEvents.UnintendedNegativeTorqueDuringOvertaking	HazardousEvent	Violated Property 12a: No SafetGoal has been derived from this HazardousEvent.
HazardousEvents.FireExplosionDuringCityTraffic	HazardousEvent	Violated Property 12a: No SafetGoal has been derived from this HazardousEvent.
SafetyGoals.AvoidUnintendedPositiveTorque	SafetyGoal	Violated Property 15: SafetyGoal has not been derived from a HazardousEvent.
SafetyGoals.AvoidUnintendedPositiveTorque	SafetyGoal	Violated Property 16: No safe state has been defined for this SafetGoal.
Functions.Generation	VehicleFeature	Violated Property 23: This VehicleFeature does not satisfy a Requirement.

Element Name	Element Type	Issue
Functions.Generation	VehicleFeature	Violated Property 0b: No descriptive Comment has been provided for this VehicleFeature.
Functions.Generation	VehicleFeature	Violated Property 0c: This VehicleFeature has not been associated with an Item.
Malfunctions.UnintendedPositiveTorque	GuidedFeatureFlaw	Violated Property 3: FeatureFlaw has no corresponding Hazard.
Malfunctions.UnintendedPositiveTorque	GuidedFeatureFlaw	Violated Property 4: FeatureFlaw has no corresponding Item.
Malfunctions.UnintendedPositiveTorque	GuidedFeatureFlaw	Violated Property 5: No descriptive Comment has been defined for this FeatureFlaw.
HazardousEvents.UnintendedNegativeTorqueDuringOvertaking	HazardousEvent	Violated Property 12a: No SafetGoal has been derived from this HazardousEvent.
HazardousEvents.FireExplosionDuringCityTraffic	HazardousEvent	Violated Property 12a: No SafetGoal has been derived from this HazardousEvent.
SafetyGoals.AvoidUnintendedP		
SafetyGoals.AvoidUnintendedP		
Functions.Generation		

Creation of a new SafetyGoal and association with HazardousEvent FireExplosionDuringCityTraffic.
Associate the SafetyGoal AvoidUnintendedPositiveTorque without HazardousEvent to the HazardousEvent FireExplosionDuringCityTraffic.

Figure 3. Violated properties are automatically identified and possible solutions are suggested on demand.

discharges the battery. If the E-motor is used as a generator to regain energy while the vehicle decelerates (recuperation), it recharges the battery and/or supplies electrical energy to the auxiliaries. These operations are controlled by an embedded system. This embedded system is clearly safety-critical since a failure of this system can cause malfunctions such as overcharging of the battery that might lead to hazards such as fire and/or explosion.

For the experimental application of the approach, PHA was carried out and an analysis model was created in course of PHA. One of the modeling elements in this analysis model is a hazardous event named *FireExplosionDuringCityTraffic*. The origin of this hazardous event is the hazard *FireExplosion* that is caused by overcharging of the battery because of unintended negative torque provided by the electric motor to the powertrain due to a failure of the safety-critical embedded system. The unintended negative torque causes the E-motor acting as a generator that finally unintendedly overcharges the battery of the vehicle.

No safety goal (top level safety requirement) has been derived from this hazardous event although it is safety-critical. This omission leads to the fact that no top level safety requirement has been derived from the hazardous event *FireExplosionDuringCityTraffic* that demands the mitigation of the hazard *FireExplosion*. Thus PHA was applied erroneously what is reflected by the analysis model. Figure 3 illustrates the property checker (1) that has detected the erroneous application based on the analysis model and reports that modeling element *FireExplosionDuringCityTraffic* violates property 12a.

Due to the erroneous application of PHA, corrective measures must be carried out and the analysis model needs to be modified. As illustrated in Figure 3, on demand the model corrector (2) proposes the creation of a new safety goal or the association with a safety goal that is already part of the analysis model. In this case the proper solution is the creation of a new safety goal and the association with the hazardous event *FireExplosionDuringCityTraffic*. Once selected, the analysis model is automatically modified and a new traceable safety goal is created. Subsequently the newly created modeling element can be refined using textual descriptions.

VIII. Conclusion

This work presents a novel approach to Preliminary Hazard Analysis (PHA) for automotive embedded systems.

The proposed framework comprises (1) an enhancement to the domain-specific language EAST-ADL, as well as (2) the identification of properties that indicate the correct application of PHA. These properties can be automatically checked based on the analysis model that reflects the results of the PHA. If properties are violated, the approach supports the automated identification of possible solutions and the automated correction of the analysis model. This guided and computer-aided approach strongly supports the application of PHA and the creation of an analysis model that properly reflects the results of PHA. The approach has been evaluated using the case study of hybrid electric vehicle development. While the use of the property checker and the model corrector did not replace the intellectual process of carrying out PHA exhaustively by a team of people with a wide variety of knowledge and skills, the automated identification of an erroneously applied PHA and the guided correction during the analysis process proved to be highly valuable to improve the quality of the analysis.

Acknowledgment

The authors wish to thank the "COMET K2 Forschungsförderungs-Programm" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support. Additionally we would like to thank the supporting company and project partner AVL List GmbH as well as Graz University of Technology. Further information about the MEPAS project can be found at http://www.v2c2.at/mepas.

References

[1] Nancy G. Leveson, *Safeware: system safety and computers*. Addison-Wesley Publishing Company, 1995.

[2] International Organization for Standardization, "ISO/DIS 26262-1 Road vehicles - Functional safety - Part 1: Vocabulary," 2009.

[3] ——, "ISO/DIS 26262-3 Road vehicles - Functional safety - Part 3: Concept phase," 2009.

[4] M. Törngren, D. Chen, D. Malvius, and J. Axelsson, "Model-Based Development of Automotive Embedded Systems," in *Automotive Embedded Systems Handbook*. CRC Press, 2008, ch. 10.

[5] D. Harel and B. Rumpe, "Meaningful Modeling: What's the Semantics of "Semantics"?" *IEEE Transactions on Computers*, vol. 37, pp. 64–72, Oct. 2004.

[6] ATESST2 Project Consortium, "EAST-ADL Domain Model Specification," 2010, version 2.1, Release Candidate 3.

[7] P. Johannessen, F. Törner, and J. Torin, "Actuator Based Hazard Analysis for Safety Critical Systems," in *Proc. of the 23th International Conference on Computer Safety, Reliability and Security*, Sep. 2004, pp. 130–141.

[8] F. Törner, P. Johannessen, and P. Öhman, "Assessment of Hazard Identification Methods for the Automotive Domain," in *Proc. of the 25th International Conference on Computer Safety, Reliability and Security*, Sep. 2006, pp. 247–260.

[9] P. Jesty, D. Ward, and R. Rivett, "Hazard Analysis for Programmable Automotive Systems," in *Proc. of the 2nd IET International Conference on System Safety 2007*, Dec. 2007, pp. 106–111.

[10] H. Schubotz, "Hazard Analysis and Risk Assessment for Complex EE-Architectures," in *Proc. of the SAE World Congress & Exhibition*, no. 2010-01-0029, Apr. 2010.

[11] M. Stringfellow, N. Leveson, and B. Owens, "Safety-Driven Design for Software-Intensive Aerospace and Automotive Systems," *Proceedings of the IEEE*, vol. 98, pp. 515–525, 2010.

[12] J. Michael, M.-T. Shing, K. Cruickshank, and P. Redmond, "Hazard Analysis and Validation Metrics Framework for System of Systems Software Safety," *IEEE Systems Journal*, vol. 4, pp. 186–197, 2010.

[13] S. Kumar, P. Ramaiah, and V. Khanaa, "A Methodology for Building Safer Software based Critical Computing Systems," in *Proc. of the 2nd IEEE International Conference on Advance Computing (IACC'2010)*, Feb. 2010, pp. 422–429.

[14] H. Giese, M. Tichy, and D. Schilling, "Compositional Hazard Analysis of UML Component and Deployment Models," in *Proc. of the 23th International Conference on Computer Safety, Reliability and Security*, Sep. 2004, pp. 166–179.

[15] K. Allenby and T. Kelly, "Deriving Safety Requirements Using Scenarios," in *Proc. of the 5th IEEE International Symposium on Requirements Engineering*, Aug. 2001, pp. 228–235.

[16] A. Sandberg, D.-J. Chen, H. Lönn, R. Johansson, L. Feng, M. Törngren, S. Torchiaro, R. T. Kolagari, and A. Abele, "Model-Based Safety Engineering of Interdependent Functions in Automotive Vehicles Using EAST-ADL2," in *Proc. of the 29th International Conference on Computer Safety, Reliability and Security*, Sep. 2010, pp. 332–346.

[17] A. Lanusse, Y. Tanguy, H. Espinoza, C. Mraidha, S. Gerard, P. Tessier, R. Schnekenburger, H. Dubois, and F. Terrier, "Papyrus UML: an open source toolset for MDA." in *Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009)*, Jun. 2009, pp. 1–4.

[18] ATESST2 Project Consortium, "Refined EAST-ADL2 tool support," Tech. Rep., 2010, Deliverable D3.2.

[19] D. Makartetskiy, D. Pozza, and R. Sisto, "An Overview of Software-based Support Tools for ISO 26262," in *Proc. of the 3rd International Workshop on Innovation in Information Technologies: Theory and Practice*, Sep. 2010, pp. 1–6.

[20] M. Ehsani, Y. Gao, S. Gay, and A. Emadi, "Hybrid Electric Vehicles," in *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles Fundamentals, Theory, and Design*. CRC Press, 2005, ch. 5.

Improving Automotive Embedded Systems Engineering at European Level

Gerhard Griessnig[1,2], Ingrid Kundner[1], Eric Armengaud[1,3], Sandra Torchiaro[4], Daniel Karlsson[5]
[1] AVL List GmbH, Hans-List-Platz 1, A–8020 Graz, Austria
[2] Institute for Technical Informatics, Inffeldgasse 16/1, A–8010 Graz, Austria
[3] The Virtual Vehicle Competence Center (ViF), Inffeldgasse 21A, A–8010 Graz,Austria
[4] Centro Ricerche Fiat S.C.p.A, Strada Torino 50, I–10043 Orbassano,Italy
[5] Volvo Technology Corporation, Götaverksgatan 10, SE–40508 Göteborg,Sweden
E-mail: {gerhard.griessnig, ingrid.kundner, eric.armengaud}@avl.com
sandra.torchiaro@crf.it
daniel.b.karlsson@volvo.com

Abstract— **Complexity in embedded systems engineering is increasing, imposing challenges to many industries. Especially the automotive industry has gone through significant changes with the application target of embedded systems moving towards safety-relevant applications. New safety standards as well as an increasing number of functionalities in a context of stringent cost constraints represent a complex challenge the industry is dealing with. In this paper, we provide an overview of two major challenges, the automotive industry is facing and of the main European research projects - with an emphasis to CESAR - focused on solving these challenges with respect to embedded systems engineering. We further discuss the reference technology platform as concept for tool integration and interoperability standard in order to significantly reduce costs. Finally, an exemplary realization of such reference technology platform for the automotive domain is presented.**

Index Terms— **automotive industry, safety-critical embedded systems, ISO 26262, integrated tool-chain**

Abstract— **Ehrhöhte Komplexität in der Entwicklung von Embedded Systems stellt heute viele Industriezweige vor Herausforderungen. Dies trifft besonders die Automobilindustrie; hier werden immer mehr sicherheitskritische Funktionen mit Hilfe von Embedded Systems realisiert. Die Wechselwirkung zwischen neuen Standards und steigender Anzahl von Funktionalitäten bei strikter Kostenbegrenzung stellt eine komplexe Aufgabe dar, die es zu bewältigen gilt. Dieser Artikel behandelt zwei dieser Herausforderungen der automotiven Industrie und stellt - mit einem Fokus auf CESAR - die wichtigsten Europäischen Forschungsprojete im Bereich Embedded Systems vor. Weiters wird die Referenz Technologie Plattform als Konzept für Werkzeugintegration und Interoperability Standard präsentiert. Ziel ist es, hiermit die Kosten des Entwicklungsprozesses spürbar zu senken. Abschließend wird eine beispielhafte Realisierung einer solchen Referenz-Technologieplattform für die Automobilindustrie behandelt.**

Index Terms— **Automobilindustrie, sicherheitskritische Embedded Systems, ISO 26262, integrierte Tool-Chain**

I. Introduction

The world market for embedded systems was strongly growing in the last decade. Nowadays, 3 billion embedded units are delivered per year and the world market for embedded systems encompasses around 160 billion Euro with an annual growth of about 9% (Ebert & Jones, 2009), (Ebert & Salecker, 2009). This trend is as well true for the automotive industry. A few decades ago, the application of embedded systems in the automotive industry was bound and mostly related to comfort features. This has changed significantly and nowadays, vehicles have been transformed to embedded computing systems with about hundred electronic control units (ECUs) and several networks running complex distributed applications.

Today's vehicles offer a high number of visible and invisible functions to the driver, to increase both the comfort as well as the safety of the passengers. Examples of such safety-related functions are anti-lock braking systems (ABS), electronic stability control (ESC) and a lot of new technologies like adaptive cruise control (ACC). Nowadays topics include hybridization, car to car and car to infrastructure communication, e-mobility etc. (Griessnig, 2011) - which leads to even more electronic functions. Their malfunction or non-function could lead to new types of accidents harming people, environment or property. To ensure that all these new functions meet a certain level of safety, new standards like the ISO 26262 (ISO, 2010) were developed, imposing requirements on the development process and the system itself.

All together, these evolutions significantly increased the complexity of car electronics and consequently a major part of the development effort is shifted to the development, design and integration of embedded systems. This evolution has led to a paradox that not only the automotive industry is facing: the complexity is increasing but costs have to be kept at a minimum while keeping the high-level quality the end-user is accustomed. This leads to high demands regarding new methods and tools, product architectures and personnel competencies. The contributions of this paper are (1) to provide an overview of the main European activities in the domain of embedded software engineering, and (2) to present the CESAR project and especially the concept of reference technology platform (RTP). The paper is organized as follow: Section II provides an overview of the challenges as well as of the main related European research projects. After that, the CESAR project and the concept of RTP are presented in

Section III. Section IV illustrates an exemplary realization of an integrated tool chain based on the CESAR RTP concept for the automotive domain. Finally, Section V concludes this work.

II. Challenges for Automotive Embedded Systems in Europe

A. Needs for Functional Safety

The automotive industry shares the view that in the next 10 years 90% of its expected innovations will be based on electrical/electronic systems with a huge emphasis on the safety related systems (Moessinger, 2010). New passive, preventive and active safety systems are added on the vehicle to decrease the probability that an accident occurs and to mitigate the consequences on vehicle occupants and other road users. The continuously increasing electrification in vehicles and the associated complexity requires a comparable level of safety within today's vehicles that takes into account the consideration of possible hazards and their impact on humans or the environment. Therefore, a safety application standard for the automotive domain has been derived from the IEC 61508 standard to consider the particular needs of this industry: the new standard ISO 26262 "Road vehicles - Functional safety". ISO 26262 represents the state of the art regarding the safety processes with the related methods and the safety requirements for the development, production, maintenance and decommissioning of E/E systems installed in series production passenger cars.

The IEC 61508 (IEC, 2010), as application independent basic standard, includes a mandatory hazard and risk assessment in order to identify the adequate safety integrity level (SIL) which is a combination of (1) consequence, (2) exposure time, (3) possibility of failing to avoid hazard and (4) probability of the unwanted occurrence. This determined SIL directly impacts the entire development process of the safety related product. One noteworthy difference between ISO 26262 and IEC 61508 is in the determination of the Automotive-SIL (ASIL as defined in ISO 26262) during the Hazard and Risk analysis (H&R). In general, the determination of the ASIL follows a similar principle where the effects of hazards are assessed in the context of possible operational situations, deriving a corresponding ASIL which is a combination of (a) severity, (b) exposure and (c) controllability. Contrary to the IEC 61508 where factor (3) can reduce the required SIL one level only, the comparable factor (c) in the ISO 26262 can fully impact the derived ASIL. In the case the hazard is "normally" under control of the driver, no ASIL independent from severity or exposure is requested.

The application of a new standard like the ISO 26262 requires changes in the development process and as well in the structure of organizations. This is costly and new techniques and methodologies that support cost-efficient development are requested to meet the ISO 26262 requirements. Such safety development techniques and methodologies provide important input for defining automated and integrated tool chains in order to preserve a stringent safety-oriented mindset for reducing development costs and the number of human errors.

B. Needs for tool-chain integration

SectionII-A emphasized the problematic trade-off between the rapid increase of new functionalities and the need to control costs, time and quality related to the development activities. Typical activities to be performed when developing new functionalities include planning, specification, modeling, analysis, verification, implementation and testing. These activities already require a significant amount of effort. On top of this, each activity further requires the use of its own set of tools with dedicated input and output formats, only loosely coupled to other tools. This complicates sharing of information between different tools, and developers are forced to manually transfer information between many different tools. An immediate consequence of this is a huge amount of double work leading to unnecessary costs. The risk of inconsistencies, which also compromise safety, between models in different tools is evident.

An additional complicating factor is routed in the separation of duties between OEMs and tier-1 suppliers. With the introduction of AUTOSAR(AUTOSAR Consortium, 2010), the responsibility split between OEMs and suppliers has changed, as now the allocation of each function is distributed among several ECUs. This implies that the OEM is responsible for a function as a whole, whereas suppliers implement well-defined parts of it. In other words, several suppliers are involved in the implementation of one function under the guidance of the OEM. As consequence, the exchange of information is not limited to different tools within one company, but can occur between different tools at different companies. The process of requirement capturing is illustrating this situation: the OEM defines a set of requirements describing the function in its own requirement management tool. These are then handed over to a supplier in a textual format (automatically generated), and the supplier needs to manually enter the requirements in his own requirement management tool. A lot of time and effort could be saved with better integrated tool chains, both between OEM and supplier as well as internally within the companies.

A first step towards this goal is the definition of common and standardized exchange formats. Two major and mutually complementary and aligned standards have been proposed for the automotive industry: AUTOSAR and EAST-ADL (The ATESST Consortium, 2010) (the latter is developed by the FP7 project MAENAD[1] with timing extensions from the ITEA2 project TIMMO-2-USE[2]). Both have defined metamodels and exchange formats with the aim to capture all engineering information needed for developing automotive embedded systems.

While the exchange formats constitute a backbone for better tool integration, the next step is to let the development tools (of both OEMs and suppliers) work directly on a common model, as defined by a standardised metamodel (e.g. AUTOSAR or EAST-ADL). A platform that controls and protects the common metamodel and orchestrates the connected tools according to a predefined development process, erases the borders between tools in the sense that it is neither necessary to model the system multiple times nor to explicitly import

[1]http://www.maenad.eu/

[2]http://timmo-2-use.org/

and export data. Such an approach allows significant savings of development effort and costs by reducing the amount of double work.

C. The ARTEMIS JU Framework

ARTEMIS Joint Undertaking[3] and national authorities are jointly funding a large number of projects in the field of embedded systems development to meet the industrial demands. The ARTEMIS JU aims to achieve effective coordination and synergy of resources and funding from the industry, the Framework Program, national R&D programs and intergovernmental R&D schemes, thus contributing to strengthening Europe's future growth, competitiveness and sustainable development (ARTEMIS Joint Undertaking, 2011). The consortia of ARTEMIS JU projects comprise industrial partners, academics, SMEs and tool vendors to bring together industrial needs, academic and industrial solutions. What is common for the projects started in this funding scheme is the focus on cross-domain solutions and sustainability of results. Examples of such projects are:

- CESAR[4]aims at improving methods, tools and processes to decrease the development effort and to introduce the reference technology platform (RTP), a seamless tool-chain for the development of embedded systems. The project and its main concepts are further explained in section III of this paper.
- iFEST[5] will, similar to CESAR with the RTP, specify and develop an integration framework for establishing and maintaining tool chains for the engineering of complex industrial embedded systems. Specific emphasis is placed on open tool chains for HW/SW co-design of heterogeneous and multi-core solutions, and life cycle support for an expected operational life time of several decades.
- MBAT that will start in June 2011, will reuse some of the CESAR results and provide Europe with a reference technology platform for effective and cost-reducing validation and verification (focus on analysis and test), focusing primarily on transportation domain, but also to be used in other domains. On the basis of models, analysis and test cases will be derived. Test cases will be used to dynamically check the system under test, analysis cases will be used to statically check the system under analysis.

The CESAR innovation cycle (figure 1) serves as an example for the cooperation between the partners: the right arrow represents the domains working in the project (automotive, avionics, automation and rail) that defined a number of pilot applications, representing current problem scenarios in their development processes. An exemplary pilot application from the automotive domain is a power train control unit for a hybrid vehicle or the recuperation function described in sectionIV. Out of these problem scenarios, requirements were formulated. The interesting challenge of this task was not only the formulation of requirements of each domain but to identify the requirements which are valid across different domains. The requirements served then as input for the work of the technical subprojects, represented by the left arrow in figure 1, in order to provide solutions to the problem scenarios identified by the domain subprojects. In the case of CESAR, the technical subprojects are requirements engineering (RE), component based devleopment (CBD) and reference technology platform (RTP). The solutions provided by these are then transferred back to the domains for evaluation based on the pilot applications.

[3]http://www.artemis-ju.eu/
[4]http://www.cesarproject.eu
[5]http://www.artemis-ju.eu/projects

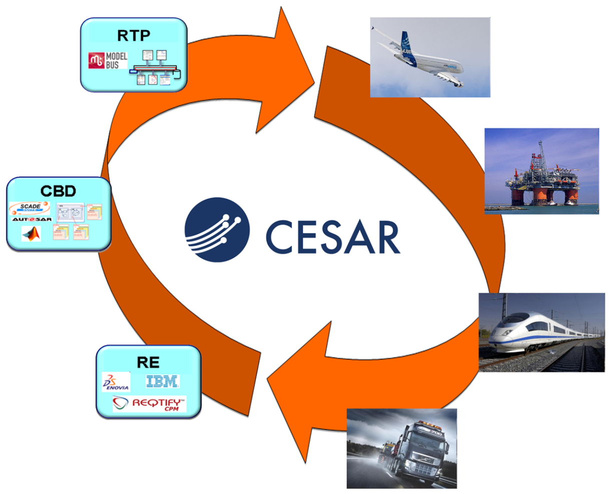


Fig. 1. The CESAR innovation cycle

III. CESAR and the Reference Technology Platform Concept

The CESAR project has been started to provide improved methods, tools and processes to meet the demands in embedded systems development in the domains avionics, automotive, automation and rail (Griessnig, et al., 2010). 55 partners from 10 countries are working for over three years in this European project that started in March 2009. The major objective is to decrease the costs of safety-critical embedded systems development by up to 50%. Corresponding to the idea of ARTEMIS Joint Undertaking, the CESAR consortium brings together partners from 4 different domains (automotive, aerospace, automation and rail) representing industry, academics and tool vendors in order to establish a significant number of stakeholders. One of the major reasons therefore is to find commonalities within the different domains and to bring together state-of-art and state-of-practice.
To meet the challenging project objectives, CESAR addresses the entire system engineering process (figure2) by improving its disciplines and integrating the innovations into a seamless tool chain, the reference technology platform (RTP). The RTP is a set of inter operable tools, methods and processes designed to improve the development work.

The following approach is taken:

1) Bringing innovations in tools and methods of the requirements engineering (RE) discipline, in particular through formalization of multi viewpoint, multi criteria and multi level requirements. The focus here is on

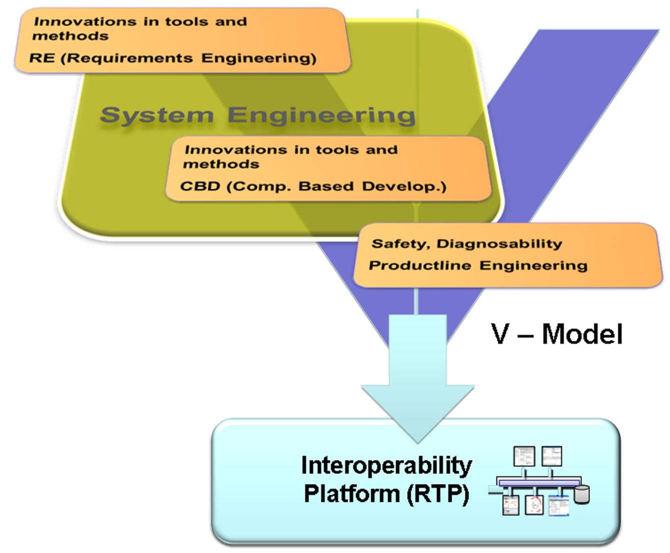

Innovations in tools and methods
RE (Requirements Engineering)
System Engineering
Innovations in tools and methods
CBD (Comp. Based Develop.)
Safety, Diagnosability
Productline Engineering
V – Model
Interoperability Platform (RTP)

Fig. 2. CESAR: The CESAR system engineering idea

the full traceability of a requirement throughout the development chain and even the entire supply chain.

2) Bringing innovations in tools and methods of the component based development (CBD) and extend it with multi view, multi criteria and multi level architecture trade offs.
3) Combining improved requirements engineering and component based development, taking into account new disciplines like safety & diagnosability (SD) and product line engineering (PLE) as a close cooperation between these is necessary.
4) Only an integration of these disciplines accompanied with an adequate tool support into a seamless tool chain (CESAR RTP) can unleash the full potential of the CESAR approach.

The realization of the RTP is a central element of the CESAR project. As shown in figure 2, it is the aim to implement the solutions provided out of the different system engineering disciplines into this seamless tool chain. It will facilitate the development of embedded systems in order to support engineers carrying out activities like tracing requirements across tool boundaries and creating deliverables e.g. safety cases. Especially in the development of safety-critical embedded systems where e.g. tracing of requirements or impact analysis are requested by corresponding safety standards, the RTP as integrated tool-chain facilitates the development work. Currently existing approaches to tool integration are either proprietary (Altheide, et al., 2003), (Ridderhof, et al., 2007) or purely academic (Burmester, et al., 2004), (Burmester, et al., 2005). CESAR however aims at finding a domain-common understanding and an agreement on concepts for tool integration between industrial partners, tool vendors and academic key players. The essential point is to move away from point to point integration to a bus-integration (e.g. ModelBus (Hein, et al., 2009)) that can be tailored to the specific demands of a development activity in the relevant domain.

Figure 3 illustrates the CESAR RTP concept with a

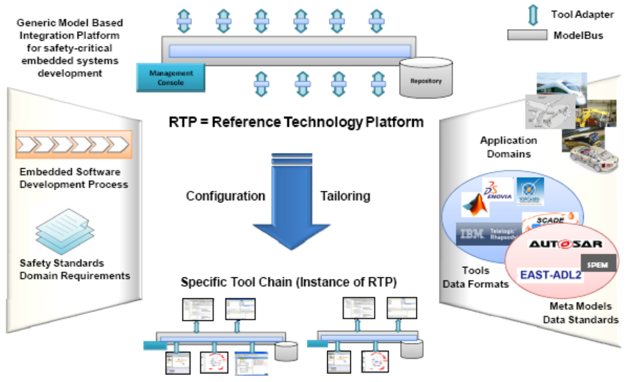

Generic Model Based Integration Platform for safety-critical embedded systems development
Tool Adapter
ModelBus
Management Console
Repository
RTP = Reference Technology Platform
Embedded Software Development Process
Safety Standards Domain Requirements
Configuration
Tailoring
Specific Tool Chain (Instance of RTP)
Application Domains
ENOVIA
SCADE
IBM Telelogic Rhapsody
AUTOSAR
SPEM
EAST-ADL2
Tools Data Formats
Meta Models Data Standards

Fig. 3. CESAR: The CESAR RTP concept

schematic representation of the RTP on the top. The instances of the RTP, the specific tool chains shown in the bottom of the picture, are created through specific tailoring rules according to relevant safety standards (e.g. ISO 26262, IEC 61508 (IEC, 2010)), application domains (e.g. aerospace, automation) or company-specific habits (e.g. preferred tools, customized processes), represented by the trapezoids left and right from the arrow.

IV. Automotive Test Management Tool-chain

Figure 4 (Armengaud, et al., 2011b) illustrates an example of a RTP tool chain, tailored to the needs of the automotive industry. The intention of this tool chain is to facilitate the interaction between different roles within the development process: the requirements engineer, system engineer, V&V manager and test engineer. The tools Papyrus[6] and AVL InMotion[7] are integrated in this tool chain over the RTP-ModelBus that assures data consistency and traceability from one development step to the next. Papyrus is an open-source tool supporting the modeling of automotive embedded systems according to EAST-ADL whereas AVL InMotion is a real-time simulation platform for maneuver and event-based testing at the testbed. EAST-ADL is an architecture description language for automotive electronic/electric systems and is tailored to support the ISO 26262 safety life cycle. The proposed tool-chain covers 7 activities to be carried out by different roles in the development process (Armengaud et al., 2011b), (Armengaud, et al., 2011a):

The first three steps are performed in the tool Papyrus according to the EAST-ADL methodology: in the first step, the requirement engineer models the requirements. The system architecture is defined in the second step by the system designer. Furthermore he adds the traceability links to the existing requirements. In the third step, the V&V manger describes the test cases. The fourth step now performs the model transformation from EAST-ADL to the AVL InMotion tool. This is done automatically by the integrated tool-chain.

[6]http://www.papyrus-uml.org
[7]http://www.avl.com, AVL InMotion powered by IPG CarMaker

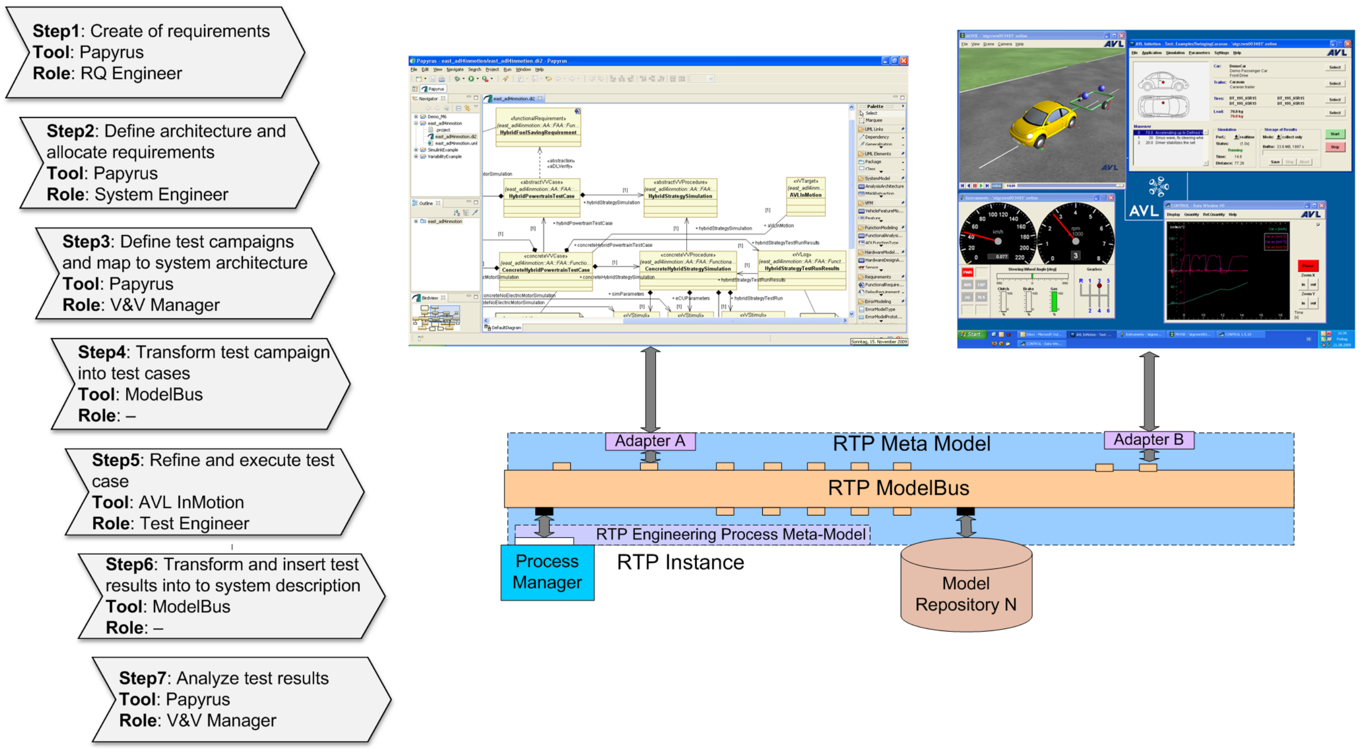

Step1: Create of requirements
Tool: Papyrus
Role: RQ Engineer
Step2: Define architecture and allocate requirements
Tool: Papyrus
Role: System Engineer
Step3: Define test campaigns and map to system architecture
Tool: Papyrus
Role: V&V Manager
Step4: Transform test campaign into test cases
Tool: ModelBus
Role: –
Step5: Refine and execute test case
Tool: AVL InMotion
Role: Test Engineer
Step6: Transform and insert test results into to system description
Tool: ModelBus
Role: –
Step7: Analyze test results
Tool: Papyrus
Role: V&V Manager
Adapter A
RTP Meta Model
Adapter B
RTP ModelBus
RTP Engineering Process Meta-Model
Process Manager
RTP Instance
Model Repository N
AVL

Fig. 4. CESAR: RTP instance for automotive test management scenario

The description of the test case done in step 3 is the input to automatically generate an InMotion test case. In the next step, the test cases are implemented in the test environment and executed by the test engineer. Step 6 transforms the results back to the Papyrus tool. During this transformation, the test execution status and a link to the test results from InMotion are automatically inserted into the EAST-ADL model. In step 7, the results are finally available and can be analyzed by the V&V expert.

In order to evaluate this tool chain, a pilot application from the automotive industry has been chosen. The subject of this pilot application is a recuperation function for a hybrid vehicle. Recuperation is the recovery of kinetic energy by the e-motor generating negative torque that decelerates the vehicle and is transformed into electric energy. This energy is further used to charge batteries or supply the low voltage board net. The safety-relevance of this pilot application is twofold (Armengaud et al., 2011a):

1) Danger of unintended recuperation leading to breaking e.g. at high speed
2) Danger of explosion or fire routed to a possible overloading of the battery

There is room for improvement in the development of electric / hybrid vehicles. Today, the passing of information between the different teams is often based on office documents under the risk of human error and a lack of traceability. What is still missing, is a harmonization of the tools and methods used, in particular for the definition of the system requirements and the design architecture.

The following benefits have been concluded from the evaluation of the proposed integrated tool chain (Armengaud et al., 2011b):

- **Fast synchronization:** Due to the use of a semi-formal language (EAST-ADL) for the system architecture, possible misunderstandings between the teams have been minimized and it allowed a quick synchronization between large teams (in our case a 15 person team across 3 countries).
- **Traceability:** A major benefit is routed in the traceability between requirements, system components, test cases and test results. This has been enabled by the meta-model covering the entire development process.
- **Easy extension of current tool-chain:** Due to the use of a vendor-neutral data backbone and a common understanding, the enhancement of the tool-chain and integration of further tools is quite easy.
- **Automated notification:** When the system has been modified and a task has to be performed, an automated notification is released. This facilitates the coordination within the team.
- **Automated configuration of subsequent development steps:** Results of one development step can be used as input for another. When using different tools, the configuration is usually done manually - leading to possible errors. Model transformation with tool adapters enables the automation of data transfer and reduces development time and human error.

(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
2. Juli 2009 (02.07.2009)

PCT

(10) Internationale Veröffentlichungsnummer
WO 2009/080384 A1

(51) Internationale Patentklassifikation:
H02P 3/06 (2006.01) ***H02P 3/08*** (2006.01)

(21) Internationales Aktenzeichen: PCT/EP2008/064104

(22) Internationales Anmeldedatum: 20. Oktober 2008 (20.10.2008)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
A2073/2007 20. Dezember 2007 (20.12.2007) AT

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **SIEMENS AG ÖSTERREICH** [AT/AT]; Siemensstrasse 92, A-1210 Wien (AT).

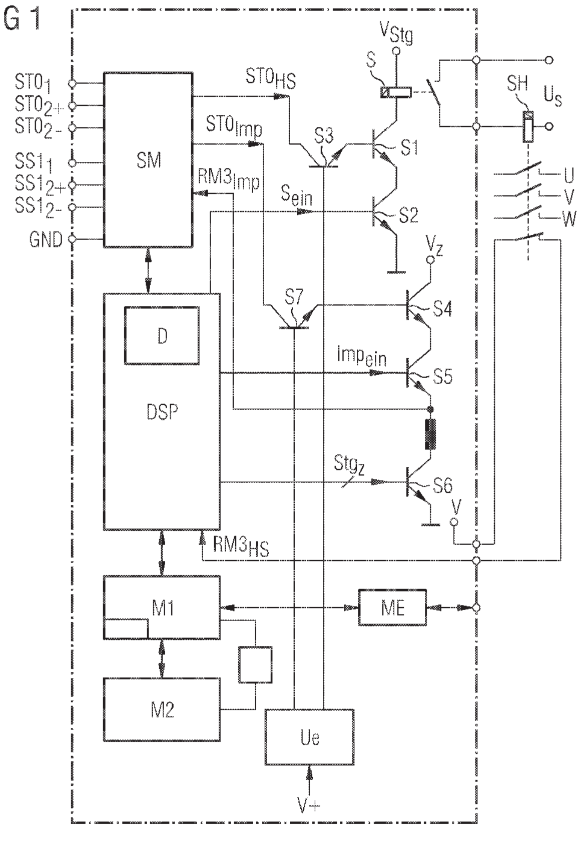
(72) Erfinder; und
(75) Erfinder/Anmelder *(nur für US)*: **FUCHS, Werner** [AT/AT]; Ziegelstadl 21, A-4890 Weissenkirchen im Attergau (AT). **GRIESSNIG, Gerhard** [AT/AT]; Brauhausstrasse 84n, A-8052 Graz (AT). **HACKL, Franz** [AT/AT]; Pacassistr. 7, A-1130 Wien (AT). **HOFMÜLLER, Wilfried** [AT/AT]; Anton-Wildgans-Gasse 3, A-2000 Stockerau (AT). **HÖRIST, Gerald** [AT/AT]; Fred-Raymondgasse 19/8/1, A-1220 Wien (AT). **MOSSBURGER, Fritz** [AT/AT]; Gustinus-Ambrosiweg 12, A-7000 Eisenstadt (AT). **SCHMID,**

[Fortsetzung auf der nächsten Seite]

(54) Title: METHOD FOR ACTUATING A DC MACHINE

(54) Bezeichnung: VERFAHREN ZUR ANSTEUERUNG EINER GLEICHSTROMMASCHINE

FIG 1



(57) Abstract: The invention relates to a method for actuating a DC machine by means of a standard controller and a safety arrangement, wherein the DC current is connected to a supply voltage by means of a main switch (HS) to be activated by a control voltage (U_s), and wherein firing pulses for actuating the electronic switch disposed in a DC converter for controlling the rotational speed and/or torque of the DC machine from a firing pulse supply voltage (V_z). For this purpose the safety arrangement is operated as an extension of the standard controller and comprises a safety module (SM), wherein an STO function for the safe torque release of the DC machine is carried out by means of said safety arrangement such that a torque release is initially provided to the standard controller upon initiating the STO function, and that the safety module (SM) does not interrupt the control voltage (U_s) via a first shut-off path of the safety arrangement and/or does not interrupt the firing pulse supply voltage (V_z) via a second shut-off path of the safety arrangement until expiration of a predetermined time span after initiation of the STO function.

(57) Zusammenfassung: Die Erfindung betrifft ein Verfahren zur Ansteuerung einer Gleichstrommaschine mittels einer Standardsteuerung und einer Sicherheitsanordnung, wobei die Gleichstrommaschine mittels eines mit einer Steuerspannung (U_s) betätigten Hauptschalters (HS) an eine Versorgungsspannung angeschlossen wird und wobei zur Steuerung der Drehzahl und/oder des Drehmoments der Gleichstrommaschine aus einer Zündimpuls-Versorgungsspannung (V_z) Zündimpulse zur Ansteuerung der in einem Gleichstromrichter angeordneten elektronischen Schalter abgeleitet werden. Dabei wird die Sicherheitsanordnung als Erweiterung der Standardsteuerung betrieben und umfasst ein Sicherheitsmodul

[Fortsetzung auf der nächsten Seite]

WO 2009/080384 A1

Roland [AT/AT]; Erdbergstr. 136/15, A-1030 Wien (AT). **SCHWARZMANN, Harald** [AT/AT]; Zwinzstrasse 7/2/4, A-1160 Wien (AT). **WAHRBICHLER, Joachim** [AT/AT]; Dr.- Eckener-Strasse 27, A-8043 Graz (AT).

(74) Anwalt: MAIER, Daniel; Postfach 22 16 34, 80506 München (DE).

(81) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare nationale Schutzrechtsart)***:** AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart)***:** ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Veröffentlicht:
— *mit internationalem Recherchenbericht*

(SM), wobei mittels dieser Sicherheitsanordnung eine STO-Funktion zur sicheren Drehmomentfreischaltung der Gleichstrommaschine in der Weise durchgeführt wird, dass bei einer Initiierung der STO- Funktion zunächst der Standardsteuerung eine Drehmomentfreischaltung vorgegeben wird und dass das Sicherheitsmodul (SM) erst nach Ablauf einer festgelegten ersten Zeitspanne ab Initiierung der STO-Funktion über einen ersten Abschaltpfad der Sicherheitsanordnung die Steuerspannung (U_s) unterbricht und/oder über einen zweiten Abschaltpfad der Sicherheitsanordnung die Zündimpuls- Versorgungsspannung (V_z) unterbricht.

2010 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems

Design and Implementation of Safety Functions on a Novel CPLD-based Fail-Safe System Architecture

Gerhard Grießnig[1], Roland Mader[1,2], Christian Steger[2], Reinhold Weiß[2]
[1]*AVL List GmbH, Austria*
[2]*ITI, Graz University of Technology, Austria*

***Abstract*—In the case of a fault fail-safe systems achieve and maintain a safe state for people, environment and property. These systems are usually realized using microcontroller-based architectures. With respect to cost per unit and development effort for fail-safe systems, industry has to consider new approaches. An option is to realize simple safety functions using architectures that include CPLDs. A novel hardware architecture for embedded fail-safe systems is the outcome of recent research efforts at SIEMENS. This architecture is homogeneously redundant and contains, in contrast to similar systems, exclusively two CPLDs instead of microcontrollers. This paper is presenting design and implementation of the very first fail-safe system based on this architecture. This system targets the market of industrial automation. The fail-safe system enhances a power converter with safety functions. To achieve the required safety integrity, adequate measures able to detect random and permanent faults, are implemented. The novel fail-safe system adheres to the draft of the second edition of the IEC 61508, which includes requirements for the realization of safety functions using CPLDs, the IEC 61800-5-2 and the EN ISO 13849.**

***Keywords*-safety; safety-critical embedded system; fail-safe system; safety function; IEC 61508; CPLD**

I. Introduction

Failures of safety-critical systems may result in death of people, pollution of the environment or destruction of property and equipment. In industrial automation, highly dependable or fail-safe systems are in use to reduce the risk of harms. In case of a failure, fail-safe systems are required to fail in a way not to harm people, environment or property. The systems have the well-defined task to achieve and maintain a safe state in case of a fault.

In domains like aviation, fault-tolerant systems guarantee the necessary availability. These systems continue to operate even if units of the systems are affected by faults. Fault-tolerant systems contain redundant units, but redundancy alone is not sufficient. Measures to detect faults, while the system is operating, are required as well. The identification of faulty units is important to identify and separate them from the faultless units to give the system the chance to reconfigure and recover.

Microcontrollers are frequently used for the realization of safety-critical embedded systems [1]. Their drawback is that they require the implementation of complex safety-integrity measures like RAM tests and CPU tests. The implementation of these safety integrity measures requires large efforts. If the safety functions, which need to be realized, are comparably simple, the biggest part of the implemented source code is dedicated to the test of RAM, CPU and periphery and only a little source code is dedicated to the implementation of safety functions.

This has been the motivation for recent research efforts at SIEMENS. The result of these efforts is a novel fail-safe system architecture [2]. In contrast to similar systems, the described system architecture contains exclusively CPLDs (complex programmable logic device) for the realization of safety functions. If simple safety functions, that require less resources have to be realized, this architecture is economically very competitive to alternate realizations based on microcontrollers. This is because the use of CPLDs that include neither CPU nor RAM makes the implementation of RAM tests and CPU tests superfluous while the costs per unit of small CPLDs is comparable to the costs per unit of a small microcontroller. Contrary, the use of CPLDs facilitates the realization of simpler safety integrity measures in hardware. This contributes to saving costs for design, implementation, verification and certification of the system.

The maturity of this concept is confirmed by a novel system based on this architecture. This system is targeting the market of industrial automation. It enhances a SIEMENS power converter which controls an electric motor with simple safety functions. Additionally, it realizes safety integrity measures to detect faults. While other contributions concern the novel CPLD-based fail-safe system architecture [2] and a method to perform fault-injection testing of CPLD-based fail-safe systems [3], this paper contributes by providing a description of design and implementation of the first fail-safe system based on this architecture.

The paper is organized as follows: Section II contains a survey of related work. Section III describes relevant standards including the draft of the second edition of the IEC 61508, which defines requirements for the implementation of safety functions in reconfigurable hardware or ASICs. The considered standards influence architecture and behavior of the presented fail-safe system as well as the development process. Section IV elaborates on the required safety functions, the architecture of the fail-safe system, its behavior and its safety integrity measures. Section V describes the applied work flow and the used tools as well as the implementation of the safety functions and the safety integrity measures in hardware. Finally Section VI concludes this work.

II. Related Work

Various architectures for safety-critical systems have been proposed so far. In [1] architectures and strategies to achieve safety integrity are described. Considered approaches are:

- **Single-controller Strategy:** A microcontroller collaborates with a primitive watchdog. The microcontroller has to reset the watchdog periodically. Additionally, the microcontroller performs self-checking diagnostics to detect faults.
- **Symmetric controller Strategy:** Two identical microcontrollers execute the same program. Their computations depend on the same inputs. The controllers communicate and compare their results to detect faults. Additionally, they perform checks for communication time-outs. Thus, each microcontroller serves as watchdog for the other microcontroller.
- **Asymmetric controller Strategy:** An intelligent watchdog (ASIC or low cost processor) collaborates with a microcontroller. The intelligent watchdog verifies the microcontroller's integrity by requesting periodic diagnostic checks.

In [4] fault-tolerant architectures are reviewed. The authors investigate how to implement these architectures, which contain multiple CPUs on a single chip. The considered architectures are lock-step, loosely-synchronized dual-processor and triple modular redundant (TMR). Additionally the authors propose new architectures for the integration on a single chip only.

The integration of fault-tolerant systems on a chip is problematic because of common cause failures. These failures affect all duplicates of a circuit due to the same reason (e.g. faulty clock tree or power supply). Thus it is difficult to reach high levels of safety integrity with this approach without additional measures.

Although the integration of safety-critical systems on a chip is difficult, an approach to reduce the probability of common cause failures is described in [5]. The authors use a library of Intellectual Property blocks, which can be used for fault detection and fault-tolerance. Their architecture includes a block which performs memory protection. A supervisor performs diagnostic checks of the CPU. The checks are mainly implemented in hardware. The system bus and the interfaces to peripheral components are supervised by separate blocks. Another bus system is used for the communication between the diagnostic units. The authors refer to this approach as faultRobust.

The approach reduces the probability of the occurrence of a common cause failure, as some blocks are architecturally or functionally diverse. Thus higher levels of safety integrity can be realized without additional measures to decrease the probability for common cause failures.

A methodology to perform a system FMEA (failure modes and effects analysis) on SoC (System on Chip) level is described in [6]. This methodology makes it possible to assess the SFF (safe failure fraction) of a SoC. The methodology has already been used to certify intellectual property, which belongs to the faultRobust approach, in adherence to the standard IEC 61508.

In [7] the utilization of PLDs to design safety-critical systems is proposed. A TMR (triple modular redundancy) system consisting of multiple PLDs is described. The authors claim that most of the faults, which affect safety-critical systems, occur at the interfaces. Thus the integration of safe input cells and safe output cells into the PLDs is proposed. A safe input cell is connected to 3 sensors, which measure the same safety-critical physical quantity. A voter circuit determines the safe input. The output of the safe output cell is dynamic. Logic level 1 is represented by an oscillating signal with a constant frequency.

Totally self-checking circuits are described in [8]. The output of circuits is separated in code words and noncode words. If no fault occurred, the output of a circuit is a code word. If the output of a circuit is a noncode word, a fault occurred. A dedicated circuit (checker) decides, if the output of a circuit is a code word or a noncode word and signals a fault if necessary.

III. Relevant Safety Standards

When a safety-critical system is developed for a certain target market, it is necessary to consider the market's demand as well as the safety standards relevant to the target market (e.g. chemical industry). The target market of the presented fail-safe system is industrial automation. There are basic standards and other derived standards that need to be considered. The presented fail-safe system adheres to these safety standards.

A. IEC 61508

The first relevant standard is the application-independent standard for functional safety. The first edition of the IEC 61508 [9], which was published in the year 2000, does not define special requirements for the development of safety related systems using CPLDs. The responsible committee for the second edition of the IEC 61508 has detected this gap. Thus the second edition of the IEC 61508, which will replace the first edition, will include detailed requirements concerning the development of safety functions using ASICs, FPGAs and CPLDs.

Currently a draft of the new version of part 2 of IEC 61508 is available which includes requirements for design, implementation and verification of CPLD-based safety-critical systems. The draft defines a V-model of the safety lifecycle for ASIC/FPGA/PLD designs. There are similarities between the development of safety-critical software and the development of ASICs. Thus the V-model of ASIC/FPGA/PLD designs is similar to the V-model of the development of safety-critical software.

Additionally, Annex B of the draft of part 2 contains a table which defines measures and techniques to avoid introducing faults in the development process of user-programmable ICs (PLD/CPLD/FPGA) for different safety integrity levels. There are requirements for design entry, synthesis, placement, routing, layout generation and production.

The presented fail-safe system is developed in adherence to the draft of part 2 of IEC 61508. Additionally, guidelines (e.g. [10]) for developing safety-critical systems using hardware description languages are considered.

B. IEC 61800-5-2

The generic standard IEC 61508 is the basis for several derived sector standards. These sector standards describe the requirements for special fields of application. In our special case, the international standard IEC 61800-5-2 [11], which describes the requirements for adjustable speed electric drive systems, has to be considered as well. To make the presented architecture applicable in our target market without any limitations, SIL 3 according to the standards IEC 61508 and IEC 61800-5-2 is aspired.

IEC 61800-5-2 requires that low demand mode of operation is not generally considered to be relevant to PDS (power drive system) applications like the power converter, which requires to be enhanced with safety functions. Therefore, PDS are considered to operate in high demand or continuous mode only. Consequently, appropriate measures have to be realized to reach SIL 3.

C. EN ISO 13849

The second basic standard concerns the safety of machinery. EN ISO 13849 [12] describes the "functional safety of machinery", which is mainly relevant to vendors of safety-critical components for machinery. This standard defines so called "categories" (Cat) and "performance-levels" (PL).

Our goal is to achieve the highest category 4 and the highest performance-level *e* to make the presented architecture applicable in our target market without any limitations. Consequently, two standard requirements defined by EN ISO 13849, need to be fulfilled:

- A single fault must not cause the loss of the safety functions.
- A single fault has to be detected at the time when a safety function is demanded or earlier.

These requirements constrain the use of an architecture consisting of two independent channels. Thus the required hardware fault-tolerance (HFT) is 1. It is not possible to satisfy these requirements with an architecture consisting of a single channel.

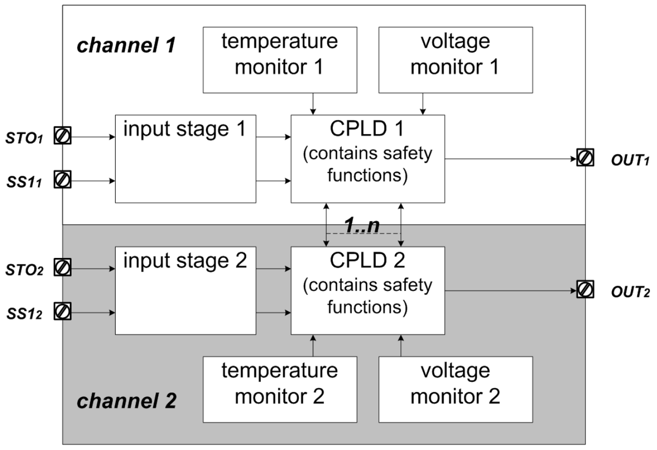


Figure 1. The Novel CPLD-based Fail-Safe System

IV. System Design

A. Safety Functions

Assume a power converter in a factory, which controls a large electric motor. This system realizes standard functions that should be able to stop this motor in different manners. If these standard functions are affected by a fault, it is not possible to stop the motor anymore. This state is unsafe as a rotating motor might cause harm to workers, the environment or the factory. Thus workers, environment and the factory are always exposed to the risk of a failing standard function.

To reduce this risk, safety functions are needed which are able to stop the motor (safe state) even if the standard functions are faulty. For two standard stop functions the corresponding safety functions, which are defined by the standard IEC 61800-5-2, have to be realized, where in this context the term PDS refers to the power converter:

- **Safe torque off (STO):** Power that can cause rotation (or motion in the case of a linear motor) is not applied to the motor. The PDS(SR) will not provide energy to the motor which can generate torque (or force in the case of a linear motor).
- **Safe stop 1 (SS1):** The PDS(SR) initiates the motor deceleration and initiates the STO function after an application specific time delay.

If the motor needs to be stopped, a standard function and a safety function are executed concurrently. Thus the system enters the safe state after a strictly specified time, even if a standard function fails. Obviously a system that implements such safety functions has to meet hard real time constraints. Thus the time that expires after a safety function is demanded and the system enters the safe state must not deviate from the defined time more than 1 ms.

The described safety functions are realized on a separate module which can be connected to the power converter. To make the fail-safe system applicable to different industrial domains, it must be possible to customize the application-specific time delay of the safety function SS1 in steps of 100 ms depending on the case of application of the fail-safe system.

B. Fail-Safe System Architecture

The presented fail-safe system is based on the novel CPLD-based fail-safe system architecture. The fail-safe system contains two channels (HFT=1). Each channel is able to perform the safety functions independently. Thus if a channel is affected by a fault, the other channel is still able to execute the safety functions. The entire system enters the safe state if a faulty channel is detected. Figure 1 illustrates the fail-safe system. It comprises following entities:

- The terminals **STO**$_1$, **SS1**$_1$, **STO**$_2$, **SS1**$_2$ are connected to an external device (e.g. a control panel) that can be used to activate the safety functions STO and SS1.
- The **input stages** represent an interface for the activation of the safety functions that assures electrical isolation by optocouplers.
- The two **CPLDs** realize the safety functions. To be able to perform diagnostic checks, the CPLDs exchange signals. The CPLDs control the safety-critical outputs (OUT$_1$ and OUT$_2$) of the fail-safe system. Each CPLD is clocked by a separate external oscillator.
- The terminals **OUT**$_1$ and **OUT**$_2$ are connected to an electric motor to allow power to be applied to the motor or to respectively shut down the motor.
- Each channel contains a **temperature monitor** to detect temperature deviations from a specified range.
- There is one **voltage monitor** for each channel which detects if the supply voltage is leaving a specified range. Furthermore, both channels of the fail-safe system are protected against dangerous overvoltage.

Since the fail-safe system constitutes a supplement to a power converter, it is directly attached to this device via dedicated pins (not illustrated). These pins are used as power supply and for the communication with a DSP that is part of the power converter and controls the electric motor. The fail-safe system and the DSP use a SPI (Serial Peripheral Interface) to communicate. This DSP is configured as SPI-master. The CPLDs are configured as SPI-slaves.

C. Fail-Safe System Behavior

When the system is in the safe state, the safety-critical outputs are switched to GND and no power is applied to the motor. Consequently, the motor coasts and is not able to cause harm to people, environment or property. If the system is in the unsafe state, the safety-critical outputs are switched to V$_{DD}$ and the motor can rotate. A rotating motor has the potential to cause harm. Figure 2 describes the behavior of the fail-safe system. In contrast to the unsafe state (MOTOR_RUNNING), the safe state can be divided

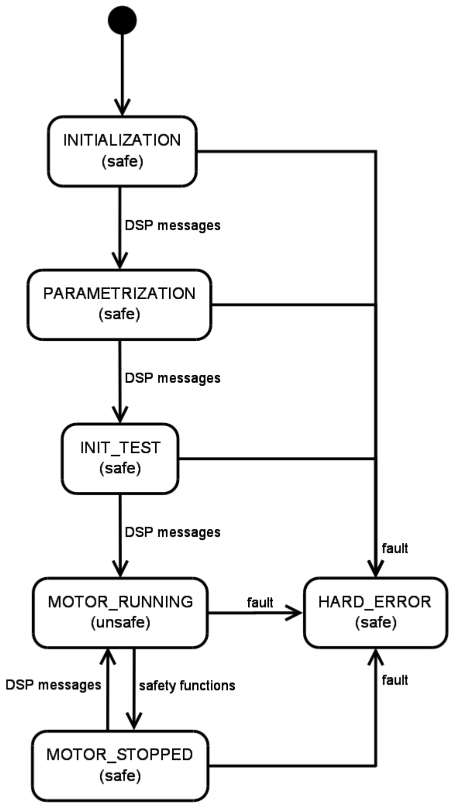


Figure 2. Behavior of the fail-safe system

into the states INITIALIZATION, PARAMETRIZATION, INIT_TEST, MOTOR_STOPPED and HARD_ERROR.

When the system is switched on, the CPLDs initialize their registers and flip-flops (INITIALIZATION). The behavior of the safety function SS1 depends on a parameter (SS1-time). Each CPLD receives a SS1-parameter from the DSP. The CPLDs store the SS1-time in volatile memory. Thus, this PARAMETRIZATION is necessary whenever the system is switched on.

When the parameterization is completed, the DSP notifies the CPLDs that a test (INIT_TEST) needs to be performed. This test verifies that the safety functions STO and SS1 are working properly.

If no fault has been detected during the init-test, the DSP tells the CPLDs that the safe state can be left (MOTOR_RUNNING).

The safety functions can be activated via the input stages. If a safety function is activated, the system enters the safe state after a specified amount of time (MOTOR_STOPPED).

If no fault occurred and the safety functions have been performed properly and are not activated anymore, the DSP tells the CPLDs to quit the safe state. In this case, the motor can rotate again (MOTOR_RUNNING).

In all states various safety integrity measures detect faults. If a fault is detected, the system enters the safe state

(HARD_ERROR). In contrast to the other states, this state cannot be left until the system is switched off.

D. Safety Integrity Measures

IEC 61508 requires SIL 3 systems with a HFT of 1 to have a safe failure fraction (SFF) of at least 90%. Also EN ISO 13849 requires a high diagnostic coverage (DC) for Cat 4, PL *e* safety functions. Thus a number of measures to achieve sufficiently high SFF and DC have to be realized, which are performed concurrently by both channels of the fail-safe system.

The fail-safe system contains two channels which are able to execute the safety function STO and SS1 independently. The channels realize the implemented safety functions by using different means. This **diversity** reduces the probability of common cause failures.

The two CPLDs exchange corresponding pairs of signals. Some of these signal pairs contain information about the state of the safety functions. If a signal pair is discrepant for a certain time, a fault can be assumed. Thus both CPLDs contain **discrepancy monitors** to signal a fault, when the discrepancy of a signal pair exceeds a specified discrepancy time. Consequently, the system enters the safe state. The discrepancy monitors check for discrepancies as long as the fail-safe system is switched on.

The used type of CPLD is similar to an FPGA. It contains look-up tables and employs channel-based routing [13]. One of the advantages of CPLDs is their "instant-on" functionality. The used type of CPLD realizes this functionality using a configuration flash memory which is loaded into CRAM-cells on start-up of the device. The **init-test** is able to detect if the safety functions can be performed properly. Thus this test can detect faults in the flash memory, the configuration or the CRAMs which affect the ability of a CPLD to perform the safety functions. This test is performed once after the parametrization of the CPLDs.

The periodic **SS1-test** starts to run as soon as the init-test was been finished successfully. The SS1-test verifies the correct functionality and parametrization of the safety function SS1. Furthermore, the functionality of the counters for the safety function STO is tested. When a number of conditions are fulfilled (expected values of counters, signals and flip-flops), the CPLDs tell the DSP that the SS1-test needs to be quit. Consequently, the DSP tells the CPLDs to quit the SS1-test. If the DSP does not respond, the system enters the safe state. Thus, the CPLDs act as a watchdog for the DSP and vice versa.

The fail-safe system can use feedback signals to observe the state of the controlled motor. If the fail-safe system enters the safe state, the motor has to stop within a specified time. The **shut down test** finds out, if this time is exceeded or not. If the specified time is exceeded, a fault is signaled. This test is detecting faults as long as the fail-safe system is switched on.

An undetected short circuit between the safety-critical outputs (first fault) is dangerous. If a second dangerous fault occurs (e.g. stuck-at-1 fault of a safety-critical output), none of the channels is able to correctly carry out the safety functions. Thus we realize a **short circuit test** which is able to detect a short circuit between the safety-critical outputs. The test is activated whenever the system enters the unsafe state and is performed when the fail-safe system enters the safe state.

Each channel contains a **temperature monitor**. If a temperature monitor detects a temperature outside the specified temperature range, the system enters the safe state.

Each channel contains a **voltage monitor**. If a voltage monitor detects that the supply voltage is too high or too low, the system enters the safe state.

V. Development Process and Implementation

A. Work Flow in Adherence to IEC 61508-2 Draft

A V-model of the safety lifecycle of ASIC/FPGA/PLD designs is defined by the draft of IEC 61508-2. It requires a clearly structured development process. Figure 3 illustrates the V-model and the names of the tools which are used for the required phases.

The result of each phase on the left-hand side of the V-model is input to the next phase on the left-hand side. In each phase, the results of the preceding phase are verified. Iterations between consecutive phases are possible. At first, the safety requirements for the E/E/PE (electric/electronic programmable electronic) system are specified and the requirements for the ASIC/FPGA/PLD are derived. For the development of the presented fail-safe system, we use the tool DOORS for requirements definition and management.

Later the architectures of the E/E/PE system and the ASIC/FPGA/PLD need to be defined. We use the tool Enterprise Architect in combination with a SysML extension to create a model of the entire E/E/PE system and the ASIC/FPGA/PLD. SysML [14] diagrams are used to specify structure (Internal Block Diagram), relationships (Block Definition Diagram) and behavior (Use Case Diagram, Activity Diagram, State Chart Diagram, Sequence Diagram). Moreover, the modules and the exact interfaces of the synthesizable VHDL-description, which defines the behavior of the CPLDs of the fail-safe system, are specified.

The definitions of the behavior and the modules follow. We create a synthesizable VHDL-description with the tool Quartus II. This tool is used to synthesize the VHDL-description and to perform place-and-route. After synthesis and place-and-route, a post-fit netlist is available. Quartus II is able to create post-fit netlist files which can be used for simulation tools and timing analysis tools.

Phases on the right-hand side of the V-model contain activities to verify the results of the phases on the left-hand side of the V-model. The post-fit netlist is verified using post-layout simulations. The draft of the IEC 61508-2

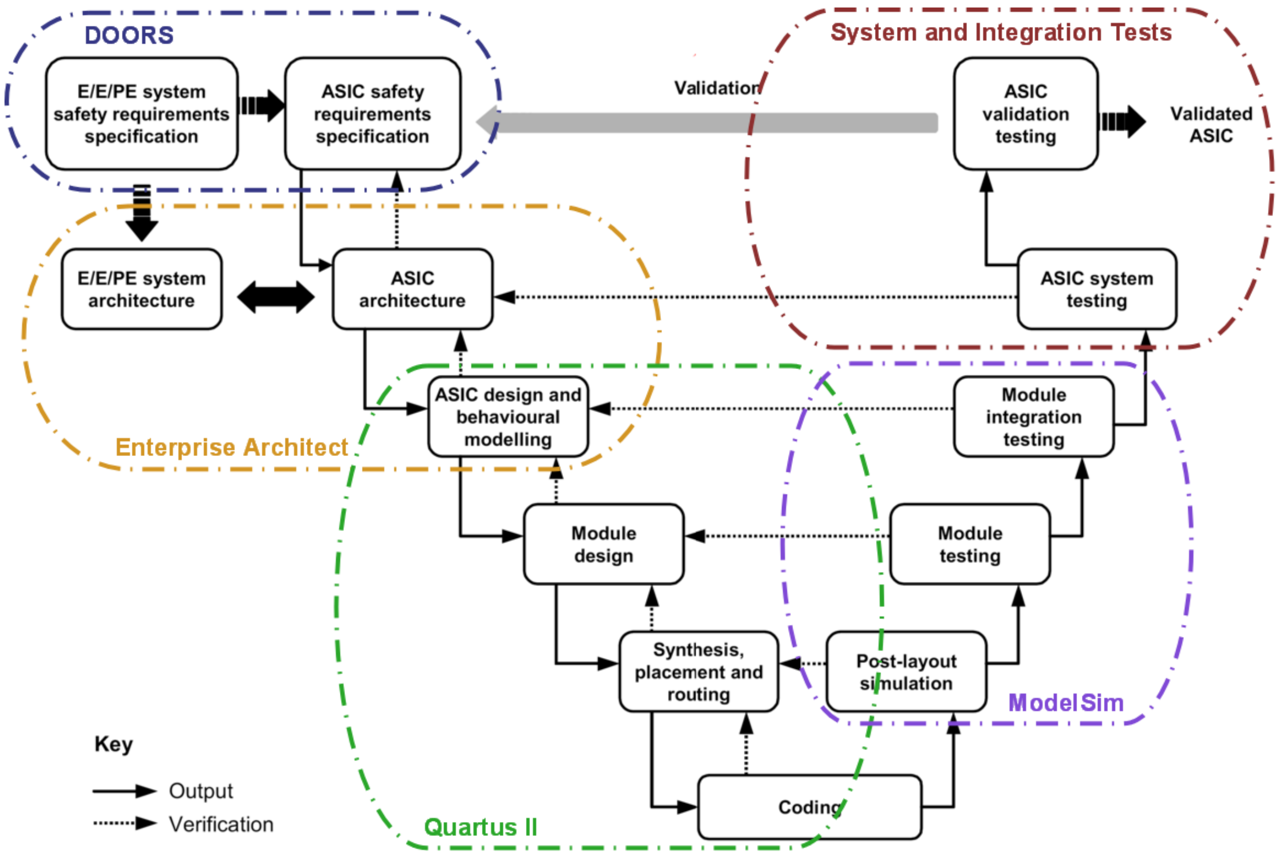

DOORS
E/E/PE system safety requirements specification
ASIC safety requirements specification
Validation
System and Integration Tests
ASIC validation testing
Validated ASIC
E/E/PE system architecture
ASIC architecture
ASIC system testing
ASIC design and behavioural modelling
Module integration testing
Enterprise Architect
Module design
Module testing
Synthesis, placement and routing
Post-layout simulation
ModelSim
Key
Output
Verification
Quartus II
Coding

Figure 3. V-model and tools

requires the alternative application of one of two equivalent techniques/measures to verify that the timing requirements for the circuit are fulfilled. We perform a static timing analysis using a timing analysis tool which is part of Quartus II. According to the draft of the IEC 61508-2, it is also necessary to apply one of two equivalent techniques/measures to verify that no systematic faults occurred during synthesis and place-and-route. We perform simulations to verify the post-fit netlist against the synthesizable VHDL-description. The tool ModelSim is used for simulation.

Module testing is required to verify the correct implementation of the modules. We create at least one test bench for each module of the synthesizable VHDL-description. The test benches are executed using ModelSim. The draft of the IEC 61508-2 requires a test coverage greater than 99% for SIL 3. We also use ModelSim to evidence that statement coverage, branch coverage, condition coverage and expression coverage reach 100% for each module.

Module integration testing verifies that the implemented VHDL-description is correct. Again, we create test benches and use ModelSim to simulate the entire VHDL-description. Some of the integration test cases are derived from use cases. Other test cases simulate the behavior of the entire VHDL-description in presence of a fault.

After the integration of the components of the E/E/PE system, we perform integration testing to verify the ASIC/FPGA/PLD. During integration testing, also fault insertion tests are performed to verify the realized safety integrity measures. To be able to perform fault insertion testing of the novel fail-safe system, it was necessary to develop a new method [3]. Finally system testing is performed to validate the entire system and the ASIC/FPGA/PLD.

B. VHDL-Description for the CPLDs

To define the behavior of the CPLDs, a synthesizable description is needed. From this description, a programming file can be created used to configure the CPLDs.

The draft of the IEC 61508-2 requires a restricted use of asynchronous constructs. Thus, the design is totally synchronous. Additionally, the draft of the IEC 61508-2 requires synthesizable descriptions to be highly modularized. Consequently, the synthesizable VHDL-description consists of various modules.

The number of required CPLD-resources is an important parameter as it directly determines the cost per fail-safe system. Thus it is necessary to create a VHDL-description which can be synthesized efficiently in terms of required CPLD-resources.

Due to the two independent channels of the fail-safe system, it is not necessary to implement resource-demanding concepts like safe input cells, safe output cells [7] or totally self-checking circuits [8] to detect faults on the CPLDs. On the contrary, the CPLDs permanently test each other to detect faults. This guarantees sufficiently high safety integrity, while fewer CPLD-resources are required. Additionally, the synthesis tool is configured to synthesize circuits optimized for low area.

Due to optimizations it is possible to implement the safety functions and the safety integrity measures using CPLDs which contain only 240 logic elements (equivalent to 192 macrocells) per CPLD. The implementation requires approximately 95% of the available logic elements. The used CPLD [13] is the smallest member of its device family.

VI. Conclusion

This paper illustrates that it is feasible to cost efficiently realize simple safety functions, using an innovative homogenously redundant CPLD-based architecture [2] for industrial automation that is in adherence to relevant standards like the IEC 61800-5-2, the EN ISO 13849 and the draft of the second edition of the IEC 61508. This draft defines requirements concerning the development of safety related systems using reconfigurable hardware or ASICs. Measures to increase the safety integrity of the system can be implemented in hardware. The presented system was assessed by an independent certification authority (TÜV SÜD). In May 2008, this certification authority reported that the presented concept is able to achieve SIL 3 in adherence to IEC 61508 and IEC 61800-5-2 as well as Cat 4, PL *e* in adherence to the EN ISO 13849, which is an important prerequisite for the certification and product launch of the fail-safe system.

References

[1] P. Sundaram and J. G. D'Ambrosio, "Controller Integrity in Automotive Failsafe System Architectures," *SAE Transactions*, vol. 115, pp. 370–377, 2006.

[2] G. Grießnig, C. Steger, and W. Reinhold, "CPLD basierende homogen redundante fehlersichere Architektur," in *Proc. of the Informationstagung Mikroelektronik (ME 2008)*, Oct. 2008, pp. 201–205.

[3] G. Grießnig, R. Mader, C. Steger, and W. Reinhold, "Fault Insertion Testing of a Novel CPLD-based Fail-Safe System," in *Proc. of the conference on Design, Automation & Test (DATE'09)*. IEEE, Apr. 2009, pp. 214–219.

[4] M. Baleani, A. Ferrari, L. Mangeruca, A. L. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-tolerant Platforms for Automotive Safety-Critical Applications," in *Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES'03)*. ACM, Nov. 2003, pp. 170–177.

[5] R. Mariani and P. Fuhrmann, "Comparing fail-safe microcontroller architectures in light of IEC 61508," in *Proc. of the 22nd International Symposium on Defect and Fault Tolerance in VLSI Systems*. IEEE, Sep. 2007, pp. 123–131.

[6] R. Mariani, G. Boschi, and F. Colucci, "Using an innovative SoC-level FMEA methodology to design in compliance with IEC61508," in *Proc. of the Design, Automation & Test in Europe Conference and Exhibition (DATE'07)*. IEEE, Apr. 2007, pp. 1–6.

[7] J. Alvarez, J. Marcos, and S. Fernandez, "Safe PLD-based Programmable Controllers," in *Proc. of the International Conference on Field Programmable Logic and Applications*, vol. 2, Aug. 2005, pp. 559–562.

[8] C. Bolchini, R. Montandon, F. Salice, and D. Sciuto, "Design of VHDL-Based Totally Self-Checking Finite-State Machine and Data-Path Descriptions," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, pp. 98–103, Feb. 2000.

[9] IEC, "IEC 61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, part 1-7," 2002.

[10] A. Söderberg, J. Hérard, and L. B. Mortensen, "Guideline for Design and Safety Validation of Safety-Critical Functions Realized with Hardware Description Language," Nordic Innovation Centre, Tech. Rep., 2005.

[11] IEC, "IEC 61800-5-2, Adjustable Speed Electrical Power Drive Systems," 2005.

[12] ISO, "EN ISO 13849, Safety of Machinery, part 1-2," 2006.

[13] P. Leventis, B. Vest, M. Hutton, and D. Lewis, "MAX II: A Low-Cost, High-Performance LUT-Based CPLD," in *Proc. of the Custom Integrated Circuits Conference*. IEEE, Oct. 2004, pp. 443–446.

[14] OMG, "OMG Systems Modeling Language (OMG SysML), V1.0," 2007.

CPLD basierende homogen redundante fehlersichere Architektur

Grießnig, Gerhard
SIEMENS AG Österreich
gerhard.griessnig@siemens.com

Steger, Christian
Graz University of Technology, Austria
steger@tugraz.at

Weiß, Reinhold
Graz University of Technology, Austria
rweiss@iti.tugraz.at

Abstract

Derzeit werden sicherheitskritische Systeme vorwiegend auf Basis von Mikroprozessoren in verschiedensten Architekturen realisiert. Aufgrund der zunehmenden Wirtschaftlichkeit von FPGAs und CPLDs in Bezug auf Stückpreise, Entwicklungszeiten und Entwicklungskosten stellen PLD basierende Architekturen mittlerweile eine Alternative dar. Hier wird auf eine CPLD basierende sicherheitsrelevante Architektur eingegangen. Zusätzlich werden die Anforderungen verschiedener Normen aus der Automatisierungstechnik berücksichtigt. Ein weiterer Schwerpunkt liegt auf einem effizienten und kostengünstigen Ansatz, die Sicherheitsfunktionen im laufenden Betrieb zu testen.

1 Einleitung

Sicherheitskritische Systeme bzw. Komponenten, die hohe Anforderungen an die funktionale Sicherheit [1] stellen, werden derzeit vorwiegend über verschiedene Mikroprozessorarchitekturen [2], [3] realisiert.

In der Automatisierungstechnik, Bahntechnik oder Automobiltechnik spielt vor allem die Fehlersicherheit (Failsafe) [4] eine wesentliche Rolle. Die Fehlersicherheit und die damit verbundene Fehlererkennung sind auch die Basis für die Verfügbarkeit von sicherheitsrelevanten Systemen [5], [6].

Einsatz alternativer Lösungen mit ASICs, FPGAs, PLDs oder CPLDs werden in sicherheitskritischen Systemen, in den zuvor erwähnten Bereichen, nur in Verbindung mit Mikroprozessoren zu Erhöhung der Diversität oder zur Erhöhung der Verfügbarkeit eingesetzt.

In diesem Paper wird auf eine homogene zweikanalige sicherheitsrelevante FPGA - bzw. CPLD - Architektur eingegangen. Ein weiterer Fokus dieses Papers liegt auf den Maßnahmen, die für das Erreichen einer hohen Sicherheitsintegrität der vorgestellten Architektur erforderlich sind.

In der Automatisierungstechnik sind für den Einsatz von Komponenten in sicherheitskritischen Systemen im wesentlichen die generische Basisnorm IEC 61508 Teil 1-7 [7], welche den Stand der Technik für die Entwicklung der funktionalen Sicherheit von elektrischen, elektronischen und programmierbaren elektronischen Systemen (E/E/PES) beschreibt und die Norm für die Maschinensicherheit EN ISO 13849 [8] von Bedeutung. Zu der oben referenzierten Basisnorm IEC 61508 gibt es weitere, je nach Einsatzgebiet abgeleitete bzw. spezialisierte, Sektornormen. Beispiele solcher Sektornormen sind IEC 61511 [9], für die Prozessindustrie, oder, die für die vorgestellte Architektur relevante IEC 61800-5-2 [10], die die Sicherheit von elektrischen Antrieben beschreibt.

Die Norm IEC 61508 definiert sogenannte Sicherheitsintegritätslevel (SIL) und die Norm EN ISO 13849 definiert Kategorien (Kat) und Performancelevels (PL), welche eingehalten werden müssen um die sicherheitsrelevanten Komponenten in entsprechenden sicherheitskritischen Systemen einsetzen zu dürfen.

Ziel ist es, hier eine entsprechende SW- und HW-Architektur von FPGAs, PLDs bzw. CPLDs zu beschreiben, mit der SIL3 bzw. Kat 4 PLe erreicht werden kann.

2 Anwendungsgebiet sicherheitsrelevante Funktionen für elektrische Antriebe

Die Motivation für den Entwurf einer hardwarenahen Sicherheitsarchitektur kommt aus der Forderung, Sicherheitsfunktionen, wie diese in der Norm IEC 61800-5-2 definiert sind, für Gleichstromantriebe zu nutzen. Hierbei soll aufgrund der zu realisierenden Sicherheitsfunktionen „Safe Torque Off“ (STO) und „Safe Stop 1“ (SS1) ein möglichst effektiver Ansatz in Bezug auf Modularität, Entwicklungszeit, Entwicklungs- und Produktkosten gefunden werden.

Die Sicherheitsfunktionen STO und SS1 bieten dem Anwender verschiedene Möglichkeiten, seinen Antrieb gemäß der definierten Risikostufe „sicher momentenfrei“ zu schalten. Um von den Industrieapplikationen unabhängig zu bleiben, muss die Sicherheitsfunktion SS1 von außen parametrierbar sein.

2.1 Risikoermittlung IEC 61508, IEC 61800-5-2

Um die Architektur uneingeschränkt für das definierte Anwendungsgebiet und für alle sicherheitskritischen, industriellen Anwendungen einsetzen zu können, gilt es die bereits oben definierte Anforderung gemäß SIL3 einzuhalten. Dabei wird, wie in IEC 61800-5-2 gefordert, vorausgesetzt, dass das System bei hoher Anforderungsrate der Sicherheitsfunktionen betrieben wird.

Sicherheits-Integritätslevel	Betriebsart mit hoher Anforderungsrate oder kontinuierlicher Anforderung (Wahrscheinlichkeit eines gefahrbringenden Ausfalls pro Stunde)
4	$\geq 10^{-9}$ bis $< 10^{-8}$
3	$\geq 10^{-8}$ bis $< 10^{-7}$
2	$\geq 10^{-7}$ bis $< 10^{-6}$
1	$\geq 10^{-6}$ bis $< 10^{-5}$

Tabelle 1. Sicherheitsintegritätslevels nach IEC 61508 bei hoher Anforderungsrate der Sicherheitsfunktion

Eine hohe Anforderungsrate einer Sicherheitsfunktion bedeutet, dass die Sicherheitsfunktion mehr als einmal im Jahr angewählt wird.

2.2 Risikoermittlung EN ISO 13849

Die zweite Anforderung kommt aus der Norm Sicherheit für Maschinen und beinhaltet die sogenannten Kategorien sowie die Performancelevels.

Die hier angestrebte höchste Kategorie 4 hat neben allen Forderungen der niedrigeren Kategorien u.a. folgende zusätzlichen Anforderungen:

- ein einzelner Fehler in jedem dieser sicherheitsbezogenen Teile darf nicht zum Verlust der Sicherheitsfunktion führen
- der einzelne Fehler muss bei oder vor der nächsten Anforderung der Sicherheitsfunktion erkannt werden, z B. unmittelbar, beim Einschalten oder am Ende eines Maschinenzyklus

Aufgrund dieser harten Anforderungen ist es nicht möglich, eine einkanalige HW-Architektur für die zu realisierende Kategorie 4 zu erstellen.

Performance Level (PL)	Durchschnittliche Wahrscheinlichkeit eines gefährlichen Ausfalls je Stunde 1/h
a	$\geq 10^{-5}$ bis $< 10^{-4}$
b	$\geq 3 \times 10^{-6}$ bis $< 10^{-5}$
c	$\geq 10^{-6}$ bis $< 3 \times 10^{-6}$
d	$\geq 10^{-7}$ bis $< 10^{-6}$
e	$\geq 10^{-8}$ bis $< 10^{-7}$
ANMERKUNG Neben der durchschnittlichen Wahrscheinlichkeit eines gefährlichen Ausfalls je Stunde, sind weitere Maßnahmen notwendig um den PL zu erreichen.	

Tabelle 2. Performancelevel PL

Der angestrebte Performancelevel e (PLe) gibt nun die Verbindung der IEC 61508 mit der EN ISO 13849 wider.

3 Architektur

Um die in Sektion 2 entsprechenden Anforderungen zu berücksichtigen, wird eine Architektur mit einer Hardware Fehler Toleranz (HFT) von 1 gewählt. Dabei werden die zu realisierenden Sicherheitsfunktionen auf zwei voneinander rückwirkungsfreien Kanälen implementiert.

Um einer angestrebten Modularität zu entsprechen, werden alle für die Sicherheitsfunktionen benötigten Komponenten auf einem eigenen erweiterbaren Failsafe-Modul (F-Modul) realisiert. Ausgenommen sind nur die passiven Bauteile, welche die eigentliche Momentenfreischaltung durchführen. Eine Anwahl der Sicherheitsfunktionen soll bei der vorgeschlagenen Architektur über Eingänge des F-Moduls möglich sein. Die Steuerung der Momentenfreischaltung soll über Ausgänge des F-Moduls erfolgen.

3.1 Normative Anforderungen an ASIC/FPGA/PLD/CPLD - Architekturen

In der derzeit aktuell gültigen Norm IEC 61508-2 gibt es Anforderungen bezüglich der HW-Entwicklung von E/E/PE – Systemen, die auch bei der Entwicklung von ASIC-, FPGA-, PLD- oder CPLD-Architekturen berücksichtigt werden müssen. Es gibt jedoch keine Anforderung an das Design, Implementierung und Tests von hardwarenahen Entwicklungen solcher Architekturen. Dies zeigt auch, dass deren Einsatz bei E/E/PE – Systemen derzeit noch keinen hohen Stellenwert besitzt.

Der Normenausschuss der IEC 61508 hat diese Lücke bereits erkannt, und versucht, diese in der zur Zeit in Überarbeitung befindlichen IEC 61508 zu schließen. Damit sollte dem zukünftigen Einsatz dieser Technologien bei E/E/PE – Systemen Rechung getragen werden.

Wie in den technischen Berichten “FPGAs in Critical Hardware / Software Systems” [11] und „Developing Critical Systems with PLD Components“ [12] beschrieben, haben bereits andere Standards, wie der RTCA DO-254 [13] oder der DefStan 00-56 [14], Anforderungen an das Design und die Entwicklung von hardwarenahen ASCI-, FPGA-, PLD- und CPLD-Architekturen gestellt.

Um den zukünftigen Einsatz der hier vorgestellten Architektur zu rechtfertigen, werden bei dieser die Anforderungen, der in Draft befindlichen, überarbeiteten Norm IEC 61508 sowie relevante Normen und Guidelines [15] für den Entwurf, das Design und die Implementierung von ASICs, FPGAs und CPLDs berücksichtigt.

3.2 Hardware - Architektur

Die Struktur der hier geforderten Architektur und der Sicherheitsfunktionen drängen einen hardwarenahen Lösungsansatz auf.

Um die Kosten pro Stück gering zu halten, wird von dem Einsatz einer kostenintensiveren FPGA Lösung Abstand genommen. FPGAs können, aufgrund der höheren Anzahl der verwendbaren logischen Elemente, zur Realisierung von komplexeren Systemen eingesetzt werden. Durch die einfache Struktur, der hier zu realisierenden Sicherheitsfunktionen, ist der Einsatz von FPGAs nicht sinnvoll. Somit kann eine kostengünstigere homogen redundante CPLD-Architektur (ca. 240 verwendbare logische Elemente je CPLD) zum Einsatz kommen.

Die im Weiteren vorgestellte CPLD-Architektur kann ohne Einschränkungen auch auf eine zweikanalige FPGA- Architektur abgebildet werden.

Die folgende Abbildung zeigt schematisch das vorgestellte F-Modul.

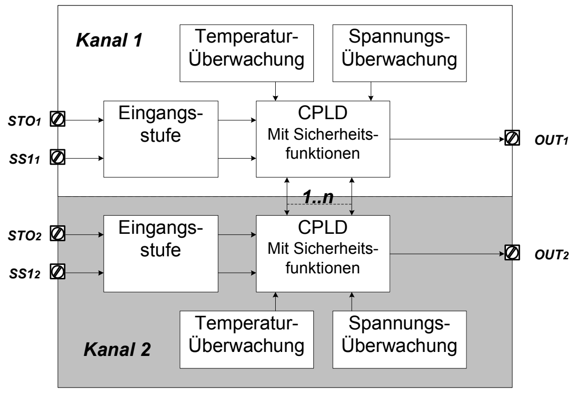


Bild 1. Zweikanalige HW-Architektur

Im Wesentlichen besteht dieser Aufbau aus folgenden, homogen redundanten Komponenten:

- Eingangsstufen: Die zwei Eingangsstufen werden für die Potentialtrennung der sicherheitsrelevanten Eingangssignale und der intern verwendeten Signale benötigt.
- CPLDs: In den CPLDs sind die eigentlichen Sicherheitsfunktionen implementiert. Diese kommunizieren für Diagnosezwecke über elektrisch entkoppelte Signale miteinander.
- Spannungsüberwachungen: Mit Hilfe der Spannungsüberwachung wird jeder Kanal separat auf Unter- und Überspannung überwacht. Zusätzlich wird das F-Modul zweikanalig gegen gefährliche Überspannungen, die zu einer undefinierten Zerstörung der Komponenten führen könnten, geschützt.
- Temperaturüberwachungen: Mit Hilfe der Temperaturüberwachung wird die spezifizierte Betriebstemperatur überwacht.

Die Spannungs- bzw. Temperaturüberwachungen sind nicht Teil der Sicherheitsfunktionen, sondern dienen dazu, die geforderte Sicherheitsintegrität der Hardware zu erreichen.

3.3 Fehlersichere „Software“ - Architektur zur Erfüllung der Sicherheitsintegrität im CPLD

Wie die Draft - Version der überarbeiteten IEC 61508-2 fordert, muss die Entwicklung von ASICs, FPGA, PLD oder CPLD, für den Einsatz in einem sicherheitsbezogenen System, in Analogie zur Entwicklung sicherheitsbezogener Software durchgeführt werden. Aus diesem Grund wird in weiterer Folge von „Software“ in CPLDs gesprochen.

Aus der obigen Forderung ergibt sich, dass Maßnahmen bereits im Design ergriffen werden müssen, damit die geforderten SIL Kenngrößen wie „Safe Failure Fraction“ (SFF), „Diagnostic Coverage“ (DC) unter Berücksichtigung des „Common Case Failure“ (CCF) und dem daraus ermittelten Wert für die „Probability of Failure per Hour“ (PFH), dem zu erreichenden SIL und der angestrebten Kat. bzw. PL entsprechen.

Da die hier vorgestellte fehlersichere „Software“ - Architektur nicht auf das Anwendungsgebiet der Automatisierungstechnik beschränkt ist, wird in weiterer Folge hauptsächlich auf die umgesetzten Maßnahmen für das Erreichen der SFF und des DC eingegangen.

3.3.1. Diversität der Funktionalität

Um den CCF, der bei homogen redundanten Systemen eine wesentliche Rolle spielt, bei den CPLDs geringer zu halten, wird eine Diversität in der HDL Funktionalität angestrebt.

Für die Entwicklung, Produktion und Fertigung ist es jedoch von Bedeutung, dass keine Diversität in der Implementierung vorliegt. Damit kann die Versionsverwaltung und der Testaufwand in der Entwicklung verringert werden. In der Fertigung können alle CPLDs mit dem gleichen Programming - File bespielt werden.

In dieser vorgestellten Architektur werden zum Beispiel die Zähler der Sicherheitsfunktionen im ersten Kanal als Aufwärtszähler und im zweiten Kanal als Abwärtszähler realisiert. Die Selektion, ob ein CPLD als Kanal 1 oder Kanal 2 arbeitet, erfolgt über einen Eingangspin des CPLD, der bereits im Layout nach logisch HI bzw. logisch LOW gezogen wird.

3.3.2. Diagnose

Um die von den Normen geforderten DC zu erhalten, werden zwischen den beiden Kanälen (siehe Bild 1) gezielt ausgewählte Signale zwischen den CPLDs ausgetauscht. Diese Signale bilden den Zustand der Sicherheitsfunktionen an verschiedenen Plätzen der Sicherheitsimplementierung im CPLD ab. Die ausgetauschten Signale werden auf jedem CPLD wieder mit den eigenen Signalen verglichen.

Um die Bauteiltoleranzen (z.B. Quarze) und das dadurch entstehende unterschiedliche zeitliche Verhalten (Timing) zu beherrschen, werden die ausgetauschten Signale auf Diskrepanzen hin überwacht. Erst bei Verletzung einer Diskrepanzzeit wird ein entsprechender Fehler gemeldet und das System in den „sicheren" Zustand versetzt sowie der sichere Zustand aufrechterhalten. Mit der Diskrepanzüberwachung wird weiters auch die Verfügbarkeit dieser Architektur verbessert, da kurz auftretende unterschiedliche Signale (z.B. EMV Störungen) nicht unmittelbar in den „sicheren" Zustand führen.

3.3.3. Selbsttests

Um permanente, zufällige oder auch transiente Fehler in der HW des F-Moduls aufdecken zu können, müssen zyklisch sogenannte Selbsttests durchgeführt werden. Die Norm IEC 61508 unterscheidet hierbei zwischen zwei Intervallen:

- Diagnosetest: Intervall, das für die Durchführung der Hintergrundtests benötigt, wird um zufällige HW-Fehler aufzudecken.
- Wiederholungsprüfung: Intervall, innerhalb dessen Wiederholungsprüfungen ausgeführt werden müssen, um gefahrbringende Fehler zu erkennen, die durch Diagnosetests nicht erkannt werden.

Da in dieser Architektur besonderes Augenmerk auf die benötigten Ressourcen und die damit verbundenen geringeren Stückkosten der CPLDs gelegt wird, müssen effiziente Selbsttests definiert werden.

Der in dieser Architektur gewählte Ansatz geht davon aus, dass nicht die Funktionalität der an der Sicherheitsfunktion beteiligten HW-Komponenten getestet wird, sondern die eigentliche Funktionalität der Sicherheitsfunktionen. Diese Tests führen beide Kanäle gegenseitig durch und sind in den folgenden Kapiteln beschrieben.

3.3.4. Hochlauftests

Bis FPGAs oder einige neueren Generationen von CPLDs (z.B. [16], [17]) betriebsbereit sind, muss zuerst die Konfiguration aus dem Flash in das Device geladen werden.

Um Fehler im Flash oder in der Konfiguration der internen Struktur eines Kanals erkennen zu können, wird in der vorgestellten Architektur ein Hochlauftest durchgeführt. Hierbei werden alle Sicherheitsfunktionen des jeweils anderen Kanals angewählt und auf Diskrepanz, zeitlichen Vergleich und Erwartungshaltung geprüft. Erst eine erfolgreiche Durchführung des Hochlauftests auf beiden Kanälen und gegenseitiger Quittierung, gibt die Sicherheitsfunktionen frei und ermöglicht das Verlassen des „sicheren" Zustandes.

Über den Hochlauftest werden die gesamten Sicherheitsfunktionen von „außen" getestet. Damit ist es möglich, bestehende Fehler im Device (Konfiguration, SRAM, …) zu erkennen, die die Sicherheitsfunktionen negativ beeinflussen.

Dieser Ansatz ist konform zu der Norm IEC 61508, welche die Annahme voraussetzt, dass ein Fehler (ausgenommen CCF) nicht auf beiden Kanälen gleichzeitig eintritt.

3.3.5. Hintergrundtests

Um permanente, transiente oder zufällige Fehler im Betrieb aufzudecken, werden Hintergrundtests durchgeführt.

In dieser Architektur wurde der Ansatz gewählt, dass solange keine Sicherheitsfunktion angewählt ist, beide Kanäle ständig die Hintergrundtests durchführen. Eine Anwahl einer Sicherheitsfunktion unterbricht den Hintegrundtest und führt die Sicherheitsfunktion aus und bringt in diesem Fall das System in den „sicheren" Zustand.

Die Hintergrundtests laufen ähnlich den Hochlauftests ab. Auch hierbei erfolgt eine gegenseitige Anwahl der Sicherheitsfunktionen, wobei der Test wiederum auf Diskrepanz, zeitlichen Vergleich und Erwartungshaltung basiert. Im Gegensatz zum Hochlauftest dürfen die Sicherheitsfunktionen nicht vollständig ablaufen, um nicht die Funktion der eigentlichen Applikation zu stören. Kurz vor Ablauf der Sicherheitsfunktion kann ein erfolgreicher Test gegenseitig quittiert werden. Erfolgt aufgrund eines Fehlers keine Quittierung, so laufen im Hintergrundtest die Sicherheitsfunktionen vollständig ab und das System geht in den „sicheren" Zustand.

Um sicherzustellen, dass die Quittierung zum richtigen Zeitpunkt stattfindet, sind beide Kanäle gegen unerlaubte Quittierungen verriegelt.

4 Resultate

Um die Konformität der hier vorgestellten fehlersicheren CPLD - Architektur zu den Normen IEC61508 1-7, EN ISO 13849, IEC 61800-5-2 und den daraus resultierenden Anforderungen für ein SIL3 und eine Kat.4 mit PLe zu zeigen, wurde diese Architektur mit den zu realisierenden Sicherheitsfunktionen STO und SS1 von einer unabhängigen, akkreditierten Zertifizierungsstelle, dem TÜV Süd [18], vorgestellt.

Ziel war es, ein offizielles sogenanntes „Proof of Concept" zu erhalten. Dieses „Proof of Concept" wurde im Mai 2008 erteilt.

5 Zusammenfassung

Mit der vorgestellten fehlersicheren CPLD - Architektur ist es gelungen, über den hier gewählten

Ansatz, nämlich die Funktion der Sicherheitsfunktionen und nicht die Funktionalität der HW-Komponenten zu testen, eine effiziente, kostengünstige und sicherheitsrelevante Variante zu erstellen.

Dieser beschriebene Aufbau berücksichtigt die Anforderungen, die für einen Einsatz sicherheitskritischer Funktionen, in der Automatisierungstechnik speziell im Bereich elektrischer Antriebe, entstehen.

Die Architektur ist mit diesem Ansatz nicht auf eine CPLD Lösung fixiert. Es kann auf rechen- und ressourcenintensive Selbsttests verzichtet werden, wenn es die Komplexität der zu realisierenden Sicherheitsfunktionen zulässt, diese von außen zu testen.

References

[1] Josef Börcsök," Funktionale Sicherheit. Grundzüge sicherheitstechnischer Systeme", Verlag: Hüthig, Auflage: 1 (Oktober 2006), ISBN-10: 3778529854

[2] Padma Sundaram and Joseph G. D'Ambrosio, "Controller Integrity in Automotive Failsafe System Architectures", 2006 SAE World Congress, Detroit, Michigan, April 3-6, 2006, *www.sae.org*

[3] Millward, J, "System architectures for safety critical automotive applications", Safety Critical Software in Vehicle and Traffic Control, IEEE Colloquium on**,** 13 Feb 1990, page(s): 4/1-4/3, London, UK, INSPEC Accession Number: 3637797

[4] Sergio Montenegro, „Sichere und fehlertolerante Steuerungen" Verlag: Fachbuchverlag Leipzig (Oktober 1999), ISBN-10: 3446212353

[5] Marcelo Lubaszewski, B. Courtois, "A Reliable Fail-Safe System", IEEE Transactions on Computers, Volume 47 , Issue 2 (February 1998), Pages: 236 – 241,Year of Publication: 1998, ISSN:0018-9340

[6] Siemens, „SIMATIC S7 F/FH Systems Projektieren und Programmieren Hochverfügbare sicherheitsrelevante Systeme", *http://support.automation.siemens.com/*

[7] Norm IEC 61508 Teil 1-7, Functional safety of electrical/electronic/programmable electronic safety-related systems

[8] Norm EN ISO 13849, Sicherheit von Maschinen

[9] Norm IEC 61511, Sicherheit in der Prozessindustrie

[10] Norm IEC 61800-5-2, Sicherheit elektrischer Antriebe

[11] Adrian J. Hilton, Gemma Townson and Jon G. Hally, "FPGAs in Critical Hardware / Software Systems", International Symposium on Field Programmable Gate Arrays, Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays, Monterey, California, USA SESSION: Poster session, Pages: 244 - 244, Year of Publication: 2003, ISBN:1-58113-651-X

[12] Adrian J. Hilton Jon G. Hall, "Developing Critical Systems with PLD Components", Foundations of Software Engineering, Proceedings of the 10th international workshop on Formal methods for industrial critical systems, Lisbon, Portugal, Pages: 72 – 79, Year of Publication: 2005, ISBN:1-59593-148-1

[13] Standard, RTCA DO-254, Design Assurance Guidance for Airborne Electronic Hardware

[14] Defence Standard 00-56 issue 2, 1997. Safety Management Requirements for Defence Systems

[15] Andreas Söderberg, Jacques Hérard, Lars Bo Mortensen, "Guideline for design and safety validation of safety critical functions realized with hardware description language", *http://www.nordicinnovation.net/nordtestfiler/rep578.pdf*

[16] Leventis, P. Vest, B. Hutton, M. Lewis, D., "MAX II: A low-cost, high-performance LUT-based CPLD", Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004, 3-6 Oct. 2004 page(s):443-446 ISBN:0-7803-8495-4

[17] Altera, "MAX II Device Family Data Sheet", *http://www.altera.com/literature/hb/max2/max2_mii5v1_01.pdf*

[18] TÜV Süd, *http://www.tuev-sued.de/home_de*

CESAR: Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems

Gerhard Griessnig[1,2], Roland Mader[1,2], Thomas Peikenkamp[3]
Bernhard Josko[3], Martin Törngren[4], Eric Armengaud[5]
[1]AVL List GmbH
[2]Graz University of Technology
[3]OFFIS – Institute for Information Technology
[4]Kungliga Tekniska Högskolan
[5]Virtual Vehicle Competence Center

Abstract

The development of industrial embedded systems is led by two a-priori contradictory constraints: minimizing the costs while maximizing product quality. This last point is especially relevant in the context of safety-critical systems where a critical product failure can harm people, environment or property and has therefore to be avoided. The management of such requirements can only be achieved with a maturing of the engineering disciplines including the improvement and standardization of the methods and development processes. This paper presents an overview and discusses the objectives of the recently started CESAR project, which goal is to improve the development process of safety-critical embedded systems. Regrouping 55 partners, this consortium moreover aims at contributing to standardization efforts from a European perspective.

1 Introduction

The evolution of embedded systems technology and applications has been occurring at a fast pace, illustrated for example through the evolution of vehicles that have been transformed to embedded computing systems with hundreds of embedded computing devices and several networks. About 3 billion embedded units are nowadays delivered per year and the world market for embedded systems encompasses approximately 160 billion Euros with an annual growth of about 9 percent [9, 10].

Embedded software combined with various hardware platforms makes it possible to deploy a larger variety of products, either in improving the performance of existing solutions or in developing entirely new products. The increased flexibility can be seen as a curse, by increasing the system complexity (design space, product configuration space and run-time state space) and at the same time as a blessing, by enabling late or even on-line fixing of bugs and addition of new features. This duality of embedded systems can be illustrated by automotive products, where embedded systems technology is an enabler for active safety systems. At the same time, however, modern vehicles are becoming partially autonomous and can, if not correctly designed, cause new types of accidents and in the worst case even harm people, environment or property.

Products which just ten years ago contained limited amounts of embedded systems now contain complex network technologies and execute several thousands lines of code. This implies that the major part of the development effort is shifted to the embedded software, testing and debugging, and system integration. Customer requirements and legislation are also evolving, and high expectations are justifiably placed on the embedded systems to be dependable and cost-efficient. Many industries are as a consequence facing a paradigm shift resulting in needs for new methods and tools, product architectures and new personnel competencies. There is an immediate need to approach the development in a more structured way in order to increase productivity, quality and time to market. Obviously, these challenges are most relevant to safety critical systems.

The CESAR project has been started in this context. This European project is funded from Artemis JU and national authorities and regroups 55 partners with a global budget of 58 M€ and a cumulated effort of 427 man-years for a duration of three years. The motivation for CESAR is to bring significant innovations in the two following system engineering disciplines: (1) requirements engineering in particular through formalization of multi viewpoint, multi criteria and multi level requirements, and (2) component based engineering applied to design space exploration comprising multi-view, multi-criteria and multi level architecture trade-offs.

The contribution of this work is twofold: First, it reviews the problems of designing and deploying safety relevant embedded systems. Second, this document discusses the approach chosen in the CESAR project to meeting the challenges. For that, the objectives are presented in Section 2. In Section 3, the innovation sub-projects are discussed. Finally, first results as well as project roadmap are provided in Section 4 and Section 5 concludes this work.

2 CESAR objectives

CESAR's main objectives are the improvement of the processes and methods for the development of safety-critical embedded systems as well as the establishment of a Reference Technology Platform (RTP), providing a conglomerate of entities, which facilitate the creation of integrated development environments for the development of safety-relevant real-time embedded systems for various domains.

These results will support the objective of reducing development time and efforts for safety-critical embedded systems development by up to 50%. Another objective is the reduction of the costs for establishment and maintenance of integrated tool chains by up to 50%. Moreover an important objective of CESAR is to find a large acceptance within the industry and technology providers (e.g. tool vendors) and to move towards an agreement on standardized integration and interoperation facilities for safety-critical embedded systems development.

An important benefit of CESAR, and a fundamental basis for the agreement on processes and methods as well as entities comprising the RTP, will be the achievement of a common understanding of basic concepts such as function, component, view, safety, and the similarities as well as differences among the domains. This overcomes the barriers and paves the way for cross-domain learning (methods, tools etc.). Similarly, the CESAR project involves several research disciplines including safety and dependability engineering, embedded systems communities including computer science and software engineering, as well as systems engineering. Bridging the gaps between these disciplines is equally important and is likely to lead to new insights.

The CESAR project is divided into seven sub-projects (SPs). While SP0 is dedicated to project management and administration, SP1 aims at the realization of the RTP that will comprise entities like meta-models, COTS, outcomes of other research efforts as well as services. It will be possible to tailor this RTP to form an integrated development environment according to a given safety-critical development process. In addition SP1 comprises the task forces Safety-Diagnosibility and Product Line. Task force Safety-Diagnosibility aims on providing appropriate and efficient approaches to the development and validation of safety-critical embedded systems. Task force Product Line is engaged in ensuring a consistent product line engineering process throughout the development cycle.

SP2 and SP3 are called innovation sub-projects. These sub-projects have been started in order to bring key innovations in the fields of requirements engineering and component based engineering. SP2 is dedicated to the development of a common requirements definition language, which is rich enough to support informal, semi-formal and fully formal requirements. The primary concern of this sub-project is ensuring traceability of requirements throughout the entire development process, the supply chain and the traceability between requirements and other design artifacts (e.g. modeling elements, test cases). The focus of SP3 is set on the adoption of component based design methodologies. This sub-project aims at facilitating system exploration on different abstraction levels, incremental verification, validation, certification and qualification and to provide tool chains to support these activities.

SP5, SP6 and SP7 are dedicated to the definition, specification and development of pilot applications from the domains of automotive, aerospace and rail and automation respectively. The term "pilot application" refers to the development of a safety-critical embedded system using a domain-specific, instantiated RTP. Each pilot application therefore provides inputs for the RTP and support its evaluation and validation, see Figure 1. Exemplary pilot applications are a power train control unit for hybrid vehicle (automotive), an engine control system (aerospace), a large safety emergency shut-down system for a petrochemical/oil-and-gas plant (automation) and a rail dispatching system (rail).

3 Project description

3.1 Reference Technology Platform

SP1 aims at developing a reference technology platform (RTP) that facilitates tool integration for the development of safety-critical embedded systems in order to support engineers in carrying out activities (e.g. requirements tracing) and creating deliverables (e.g. safety case). Building the RTP consists of the following main steps, where this chapter focuses on the first two points:

Implementation of the RTP will be performed by (1) integrating existing standards of meta-models (for architectures, components, requirements...) that have been developed and successfully applied in the past, (2) definition of a process engineering meta-model including specification of how (existing and new) tools, meta-model entities, basic services etc. are to be used during a specific design activ-

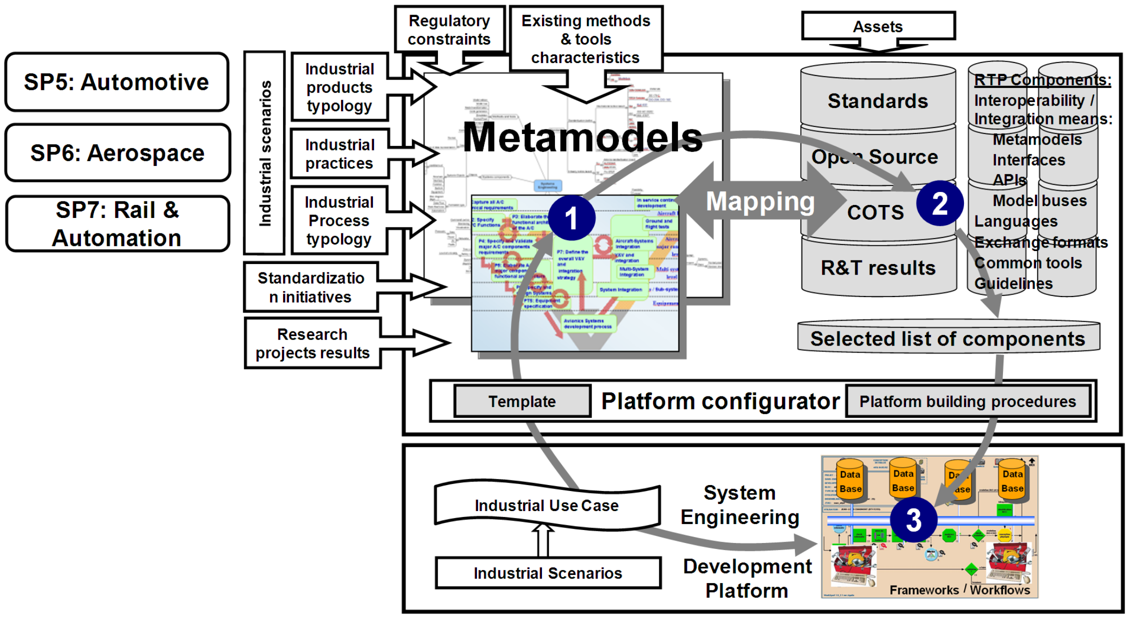


Figure 1. CESAR: Project Overview

ity plus interoperability specification and guidelines for the instantiation of the RTP in a given design context, (3) definition of processes and associated means how to specify, implement, and maintain the RTP's components with regards to safety concerns, (4) ensure that safety and diagnosability requirements are taken up in compliance with the relevant standards in respective domains, (5) supporting a consistent Product Line Engineering process throughout the development cycle, and (6) providing dissemination and training material for the instantiation and application of the RTP.

Existing approaches to tool integration for safety-critical embedded systems development are proprietary [3, 19] or purely academic [7, 6]. In opposite CESAR aims at achieving a common understanding and an agreement on concepts for tool integration for safety-critical embedded systems development between industrial and academic key players in the domains automotive, aerospace, automation and rail. Moreover CESAR aims at a flexible approach where both, solutions of vendors and SMEs as well as open-source solutions can compete in offering best-in-class solutions for particular stages of these processes. This approach will be based on already identified (and yet to be identified) cross-domain commonalities in the development processes for safety-critical embedded systems.

Achieving the acceptance of industrial stakeholders and tool vendors requires the ability of the RTP to integrate *existing* tool frameworks, often equipped with their own metamodel, as well as requirement and configuration management facilities. Thus, the implementation of the RTP is strongly driven by the need *to identify* common aspects in these environments, to provide a consistent view based on them, and to make them available to design, analysis and validation methods. This will be facilitated by the development of a proper RTP meta model. Thus CESAR fully considers the strong and ongoing trend towards meta-model based tool integration of recent research activities [15].

The main concept of the RTP is to provide a conglomerate of entities that can be (automatically) combined in different ways in order to provide a seamless development environment (RTP instance) tailored to the product to develop, see Figure 2. The entities, (top of Figure 2) can either be approved COTS, innovative outcomes of research projects, meta models, tool integration frameworks or supporting services. The subset of required entities can be defined using tailoring rules according to relevant safety standards (e.g. ISO WD 26262, EN 50128), application domain (e.g. automation, aerospace) or company-specific habits (e.g. preferred tools, customized processes). Depending on these tailoring rules process-driven tailoring of the RTP (middle of Figure 2) can be performed. The tailoring step comprises the generation of a customized integrated tooling environment tailored to relevant safety standards, application domain and company habits. This generation comprises activities like tool installation, deployment of adapters for the integration of tool-specific meta-models, and instantiation of a process manager according to company-specific process needs.

The results of the tailoring step is the RTP instance (bottom of Figure 2). It consists of the different modeling tools as well as the RTP-Modelbus (based on [12]) in charge of the communication between these tools. The RTP instance thus efficiently supports model based development (see Section 3.3), which is based on the continuous model refinement of the product being developed: each tool is used to perform a refinement step, and the Modelbus assure data consistency from a development step to the next one. A dedicated RTP Meta-Model is used in order to avoid proprietary solutions and assure interchangeability between the tools. Connections to the RTP-Modelbus will be either realized via adapters for converting tool-internal model representations in the RTP Meta-Model, or directly integrated to the RTP Meta-Model. Analysis tools (such as [16]) will be connected in the same way. A similar connection will also be possible for managing the system engineering process. In this case it is not sufficient to represent the elements of the system under development (SUD) in a standardized form, but also the artifacts that are needed for the definition of this process. Such primitives are provided by the RTP Engineering Process Meta-Model and will play an important role when the platform needs to be tailored for the different application domains. The function of the repository is to maintain the different tools in sync so that modifications of the model by tool X become visible by tool Y and vice versa. In addition, the repository is able to provide storage of model artifacts (for tools without corresponding facilities).

At present, several guidelines for the development of the RTP have been identified that satisfy needs that are common to nearly all safety-relevant standards, e.g.: it must be possible to associate all elements of the SUD with relevant V&V artifacts (requirements, implementation status, testcases, analysis results, simulations, results from tests and experiments); for all the elements of the SUD (i.e. not only the system) it needs to be possible to associate a boundary and boundary conditions with them; it must be possible to represent the evolution of the system architecture (including functional, software, and various hardware aspects) based on the elements of the SUD; the status of the V&V process must be traceable to the system architecture and it must provide the relevant traceability links based on V&V artifacts associated with the architectural elements.

3.2 Requirements Engineering

Operational and performance requirement engineering and management are one of the decisive pre-conditions to establish efficient certification, safety and development processes. Novel approaches to establish seamless tool chains are needed to reduce the most crucial cost drivers when establishing safety cases: the extensive documentation needed for certification and the evidence gained from rigorous V&V-related processes.

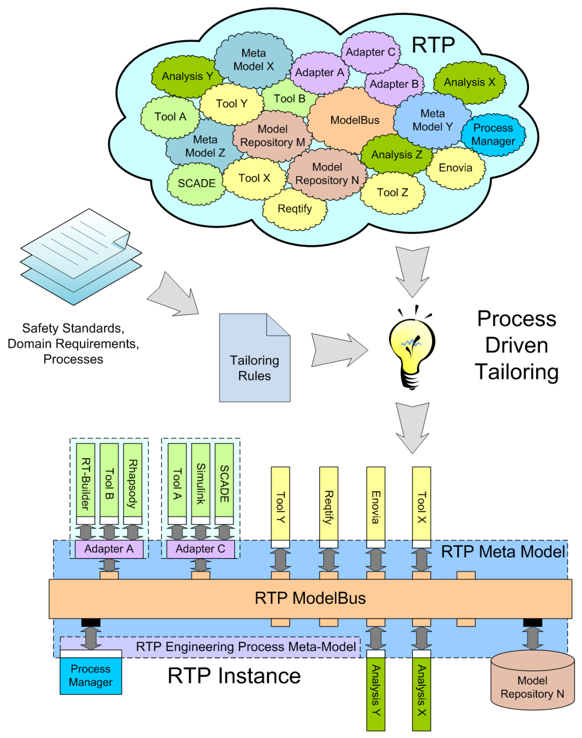


Figure 2. RTP: Tailoring & Structure

Despite obvious progress through model-based engineering techniques and especially requirement engineering tools, industrial practice in general lacks seamless solutions to cope with the most critical development challenge: consistency in the case of changes along the chain of development artifacts from requirements to code and tests. Up to now traceability, relying on breakdown, refinement, assignment and implementation of requirements, ensured by verification and validation, remains a largely manual, thus error-prone, time consuming and costly activity.

Whereas functional requirements are well managed in practice, extra-functional requirements such as RAMSS (reliability, availability, maintainability, safety, security) are a specific concern. Despite recognition of these requirements as crucial for the success of products, the key issues have not been mastered. Breakdown and assignment, tracking and early verification/analysis through continuous development stages are especially difficult. In the end, traditional testing is not the appropriate means to establish trustworthy evidence throughout the development until final validation.

Hence the objective of SP2 is to improve the current processes dealing with requirements management and engineering in order to favor interchange in the development and supply chain and to ease the definition and identification of safety critical requirements in line with safety standards. Supported by traceability mechanisms, verification and validation of requirements shall be performed through improved techniques and methods. From a technical point of view the aim is:

- To provide a formalized multi criteria requirement capturing language (called RSL) together with a tool independent exchange format.
- To ensure complete traceability and consistency of requirements from concepts to products across supplier-chain boundaries.
- To support validation of formal multi-criteria requirements with respect to consistency and completeness.

A precondition of automated verification and validation of requirements is that these requirements are given a formal notation. Pure formal languages as first or second order logic, Z [21], B method [1], temporal logics [17, 11] and other approaches are from a mathematical / theoretical point of view well defined with a clear and unambiguous semantics which allows formal analysis to be applied. Such pure formalisms are rarely (if at all) accepted in industry. The main reason is that the syntax is typically not user friendly and the semantics is often hardly accessible. A user friendly notation with some syntactical sugar is needed (e.g. PSL[1] [5] provides some syntactic sugar on top on temporal logic) to overcome these problems.

In order to address several levels of abstractions – high level requirements which are mainly given in textual form down to technical requirements on a detailed design level which may be specified in a formal notation – and to cover several aspects – functional as well as extra-functional requirements – the RSL will have several facets including free text, structured text, and model-based approaches.

Due to the increased system complexity of integrated embedded systems and due to the involvement of several engineering teams – also across company borders – an improved requirements engineering process is necessary to fulfill the requirements regarding costs, quality, safety standards, and certification (DO178B, EN50128, ISO WD 26262, IEC61508 and others). To deal with the management and traceability of requirement we will develop a Requirement Engineering framework built on existing concepts as established in SysML[2], EAST-ADL2[3], meta-models from the MeMVaTEx project [2] or in the traceability reference model by Jarke and Ramesh [18].

The process model combined with the requirement specification language will provide enhanced requirements management methods supporting traceability across the complete development process, across the supply chain, between requirements and modeling elements and derived design artifacts, and allows to introduce enhanced methods ensuring requirements V&V compliant to standards:

- enhance and measure completeness of functional and extra functional requirements,
- ensure consistency and soundness of functional and extra functional requirements,
- requirements-based testing for functional and extra functional requirements,
- establish Code of Practice for safety-critical applications.

3.3 Component-based development

SP3 focuses on system architecture and system detailed design, relying on component based design. While safety-critical systems can be developed today, this is very costly and reuse is difficult because of the lack of appropriate methods, tools and processes. Some of the aspects that are studied in SP3 include the integration of multiple views and multi-criteria in design space exploration, and incremental approaches for verification and validation. While the needs to support multiple views, such as for example, functional, hardware, safety and performance viewpoints are widely recognized and many multi-view frameworks have been proposed (see e.g. [20] or IEEE 1471[4]), the integration of the views and information management still pose challenges. The goal is to deliver improved methods, tools and processes with particular consideration of product safety and performance, and the fact that the products are developed as product lines.

The CESAR project relies on advances made in several previous projects including major initiatives such as

- The EAST-EAA, ATESST, and ATESST2 projects leading to EAST-ADL architecture description language;
- TOPCASED leading to a comprehensive open-source environment for safety-critical system design;
- The SETTA, ESACS, ISAAC and ATESST2 projects in achieving coherency between system development models and safety analysis models through model-based safety analysis techniques;
- The SPEEDS[5] project developing multi-criteria contract based design and semantic based tool interoperability standards through its SysML compliant meta-model of Hierarchical Rich Components [13].

[1]http://www.eda.org/vfv/docs/PSL-v1.1.pdf
[2]http://www.omg.org/spec/SysML/1.1/
[3]http://www.atesst.org
[4]http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html
[5]http://www.speeds.eu.com/

Alignment with domain standards is central, including architectural standards such as Autosar[6] and IMA, and domain-specific safety standards such as ARP 4754, IEC 61508 and ISO WD 26262. Also international domain independent standards from OMG[7] and INCOSE are important to consider.

Key tenets of SP3 include the fusing of the following approaches:

- Model-based and component-based development approaches, MBD and CBD [22]
- Model-based embedded systems development and safety engineering

Model based (or model-driven) development emphasizes the use of higher levels of abstractions (of functions and hardware), and the use of model-transformations to create specific models (views) at the same level of abstraction, and for bridging levels of abstraction (basically operating as a kind of compilers). While tool-chains supporting functional modeling and code generation are already well established in several domains, architecture modeling has been lagging behind.

Recent developments in architecture description languages (ADL's), in particular Autosar, AADL[8] and EAST-ADL, have changed this situation and make it possible to extend the scope of MBD. ADL's focus on describing the system structure, mainly with the notion of black box (SW/HW) components. They thus follow the lines of compositionality where the idea is that system properties can be derived from a configuration of components and their externally visible properties. This situation fits well for the purposes of a system integrator that specifies the system architecture and has to reason about system level properties without details of the internals of component implementations.

As an example of a language with multiview modeling capabilities, the EAST-ADL2 extends traditional ADLs by defining a system model that organizes the engineering information in multiple models, reflecting different views and levels of details of the embedded system architecture. From a system development perspective, other ADL's such as AUTOSAR and the AADL cover only one level of abstraction of a system architecture, the implementation level.

The EAST-ADL2 language provides an integrated system model with traceability through the different levels. The language supports model-based architecture analysis of a complete embedded system through the formalization of system structure and properties. Special emphasis has also been placed on modeling support for assessments of timing and dependability. The EAST-ADL is harmonized with Autosar, the ISO WD 26262, MARTE and implemented as UML2 profile.

Component based development (CBD) derives from software development and middleware with the intention to provide explicit definitions of component interfaces and interactions. Given more explicit definitions of components and their intended contexts, reuse is facilitated. In a typical CBD environment, configuration of components is followed by glue code generation for a particular platform. When combining MBD and CBD, components, and component models, will exist and have to be defined at several levels of abstraction. The more stringent definition of components facilitates reuse as well as predictable composition of components. The gap between safety engineering and model-based development has been addressed by previous projects, for example SETTA [14], but through point-to point integration efforts. The goal in CESAR is to accomplish a systematic integration that supports adequate safety processes, see [8] for one example effort addressing this for automotive embedded systems.

Sub-project 3 in CESAR is structured into four main activities as follows:

- Modeling languages and validation techniques: This activity addresses the MBD challenge of defining appropriate methodologies to deal with the required types of models (the system environment, functions, architecture, safety aspects), and their relations, including different levels of abstraction.
- Architecture design and design space exploration: Embedded systems have to fulfill many requirements including functionality, performance, safety, maintainability and solvability. These qualities are conflicting and related by shared resources and design decisions that affect more than one quality [4]. While there still is a need for fundamental research on this topic, CESAR will tackle this topic by enhancing multi-view modeling, trade-off analysis, and by attempting to reconcile existing methods and processes.
- Component based detailed design and validation, with the objective to specify a methodology with consideration of certification and qualification.
- Tool specification and tool implementation, with the objective to develop tool chains (enhancements of existing as well as new) that will be made part of the RTP.

4 CESAR Roadmap and first results

The CESAR project represents an exception with regards to budget volume, efforts, number of participants and large scope of research topics addressed. In order to ensure the realization of the ambitious goals, the project is divided into

[6]http://www.autosar.org
[7]http://www.omgmarte.org/
[8]http://www.aadl.info/aadl/currentsite

three similar phases, see Figure 3. This iterative development aims at providing a first version of the RTP only nine months after project start, and periodically improving this platform according to the experience gained from the different application domains.

Each iteration consists of the evaluation of the current state of the art (research point of view) and state of practice (industry point of view), which also includes the evaluation results of the last iteration. Parallel to that, the pilot applications are specified or refined and the requirements collected. Then, these domain requirements are used as inputs for the sub-projects RTP (SP1), requirements engineering (SP2) and model based engineering (SP3) for the innovation cycle. Finally, the different pilot applications are correspondingly enhanced and the methods and processes are evaluated for the next iteration.

There are four major milestones in the CESAR project. The first milestone (November 2009) represents the establishment of the technical basics required to achieve the ambitious goals of CESAR. The second and third milestones (respectively December 2010 and October 2011) are defining the end of the innovation cycles. The fourth milestone is the project end (February 2012). Additionally, a business model is generated in order to develop a self-sustaining ecosystem (e.g. including standardization of the RTP) guaranteeing the existence of the RTP after the project end.

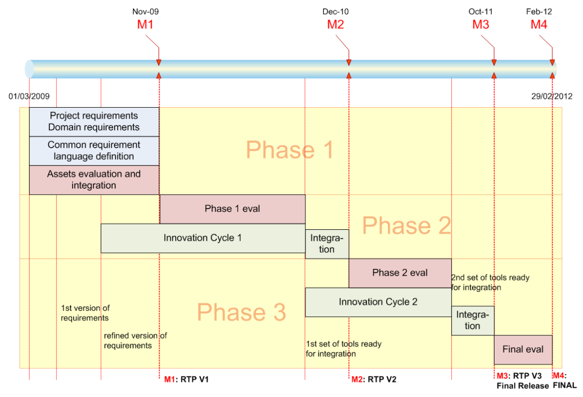


Figure 3. CESAR: Roadmap

Currently, demonstration scenarios and experimental prototypes are developed for the aerospace and the automotive domain with the intention to proof the technical concept and to support the specification of the RTP.

Exemplary, the automotive scenario includes the connection of the tools Papyrus[9] and AVL InMotion[10] using the RTP-modelbus presented in Section 3.1. Papyrus, on one side, is an Eclipse-based open-source tool that supports the modeling of automotive embedded systems according to EAST-ADL2. AVL InMotion, on the other side, is a real-time simulation platform for maneuver and event based testing at the test bed. It supports key business objectives such as hybridization and electrification of power train engineering.

This scenario illustrates how contemporary tool integration infrastructure can be used to facilitate multi site, multi user collaboration and to establish a tracing between the artifacts of the project like requirements, test cases, stimuli and test results. This exemplary tool chain allows to carry out different activities such as (a) system modeling, (b) mapping of system requirements to test cases, and (c) definition and execution of test cases. While the two first activities can be performed within one environment (Papyrus), the definition and execution of test cases is performed within a totally different environment (in this example AVL InMotion) by different roles. Efficient information exchange is required both regarding the data (e.g. which configuration/boundaries should be tested) and the control flow (e.g. when a new test case is available or has been completed). More precisely, the development process (control flow) has to be defined and integrated within the tool chain, and a common meta-model for efficient data exchange between the tools is required, see Figure 4. More especially, a mapping is required between the EAST-ADL and the AVL InMotion meta-models in order to support this tool integration and transformations between models.

The advantages of this quite simple scenario are (1) efficient cooperation between different expert teams using well defined processes, and (2) the improved traceability between models, requirements and test cases. When extrapolating these results to real development environments (including much more tools, methods and processes), it is evident that the proposed CESAR approach will strongly improve the development quality and efficiency of embedded systems.

5 Conclusion

The CESAR project aims at providing a solid foundation for the efficient development of safety relevant embedded systems. Regrouping major European companies and research institutions, this multi-domain approach will both provide technical answers and be an important cornerstone toward the establishment of a de-facto industry standard.

The approach relies on the development of a customizable system engineering "Reference Technology Platform" (RTP) that supports multi-viewpoint based development process (for support of functional and non-functional aspects), multi-criteria based design processes (for optimization of designs to multiple objectives functions) and multi-level design flows (to cover all stages from initial concepts

[9]papyrus-uml.org
[10]www.avl.com

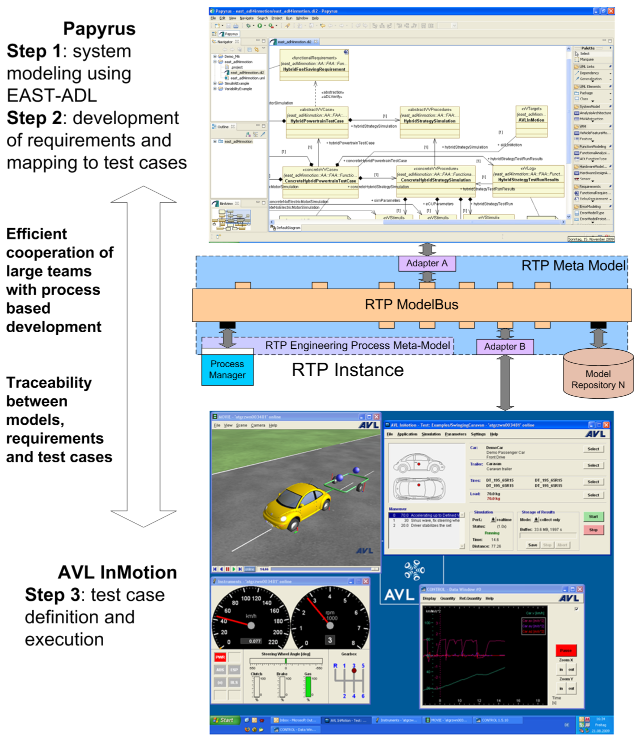

Papyrus
Step 1: system modeling using EAST-ADL
Step 2: development of requirements and mapping to test cases
Efficient cooperation of large teams with process based development
Traceability between models, requirements and test cases
AVL InMotion
Step 3: test case definition and execution
Adapter A
RTP Meta Model
RTP ModelBus
RTP Engineering Process Meta-Model
Adapter B
Process Manager
RTP Instance
Model Repository N

Figure 4. CESAR: Demonstration scenario

to design and implementation). This aims at moving toward "first-time-right" designs and making the development process more efficient .

Acknowledgment

The authors would like to thank all colleagues from the CESAR consortium for their contribution and acknowledge the productive ongoing collaboration. The complete list of project partners (unfortunately too long for this section) is available at the CESAR website[11]. The authors also want to thank Artemis Joint Undertaking and the corresponding national authorities for funding the CESAR project (contract number 100016). The opinions presented here reflect only the authors' views and the Joint Undertaking is not liable for any use that may be made of the information contained therein.

References

[1] J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] A. Albinet, S. Begoc, J.-L. Boulanger, O. Casse, I. Dal, H. Dubois, F. Lakhal, D. Louar, M.-A. Peraldi-Frati, Y. Sorel, and Q.-D. Van. The MeMVaTEx methodology: from requirements to models in automotive application design. In *ERTS'08, Toulouse, France*, 2008.

[3] F. Altheide, S. Dörfel, H. Dörr, and J. Kanzleiter. An architecture for a sustainable tool integration. In *Proc. of the Workshop on Tool Integration in System Development, European Software Engineering Conference (TIS 2003)*, pages 29–32, 2003.

[4] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison Wesley, 1998.

[5] I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic sugar. In *13th International Conference on Computer Aided Verification 2001*, pages 363–367. Springer, 2001.

[6] S. Burmester, H. Giese, M. Hirsch, D. Schilling, and M. Tichy. The fujaba real-time tool suite. In *Proc. of the 27th International Conference on Software Engineering (ICSE 2005)*, pages 670–671, 2005.

[7] S. Burmester, H. Giese, J. Niere, M. Tichy, J. Wadsack, R. Wagner, L. Wendehals, and A. Zündorf. Tool integration at the meta-model level: the fujaba approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6:203–218, 2004.

[8] D. Chen, R. Johansson, H. Lönn, Y. Papadopoulos, A. Sandberg, F. Törner, and M. Törngren. Modelling support for design of safety-critical automotive embedded systems. In *The 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP 2008)*, pages 72–85, 2008.

[9] C. Ebert and C. Jones. Embedded software: Facts, figures, and future. *IEEE Computer*, 42(4):42–52, 2009.

[10] C. Ebert and J. Salecker. Embedded software technologies and trends. *IEEE Software*, 26(03):14–18, 2009.

[11] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[12] C. Hein, T. Ritter, and M. Wagner. Model-driven tool integration with modelbus. In *Proc. of the Workshop Future Trends of Model-Driven Development*, 2009.

[13] B. Josko and A. Metzner. Designing embedded systems using heterogeneous rich components. In *Proceedings of the INCOSE International Symposium*, 2008.

[14] Y. Papadopoulos, J. A. McDermid, R. Sasse, and G. Heiner. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure, reliability engineering and system safety. *Elsevier Science*, 71(3):229–247, 2001.

[15] R. Passerone, I. Ben Hafaiedh, S. Graf, A. Benveniste, D. Cancila, A. Cuccuru, S. Grard, F. Terrier, W. Damm, A. Ferrari, L. Mangeruca, B. Josko, T. Peikenkamp, and A. Sangiovanni-Vincentelli. Metamodels in europe: Languages, tools, and applications. *IEEE Design and Test of Computers*, 26(3):38–53, 2009.

[16] T. Peikenkamp, A. Cavallo, L. Valacca, E. Böde, M. Pretzer, and E. M. Hahn. Towards a Unified Model-Based Safety Assessment. In *SAFECOMP*, pages 275–288, 2006.

[17] A. Pnueli. The temporal logic of programs. In *IEEE Symposium on Foundation of Computer Science*. IEEE Computer Society Press, 1977.

[11] http://www.cesarproject.eu

[18] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEEE Trans. on Software Engineering*, pages 58–93, 2001.

[19] W. Ridderhof, H. G. Gross, and H. Dörr. Establishing evidence for safety cases in automotive systems - a case study. In *Proc. of the 26th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2007)*, pages 1–13, 2007.

[20] R. Siegers. The abcs of afs: Understanding architecture frameworks. In *Proceedings of the INCOSE International Symposium*, 2005.

[21] J. M. Spivey. *The Z Notation: a reference manual*. Prentice Hall, 1992.

[22] M. Törngren, D. Chen, and I. Crnkovic. Component based vs. model based development: A comparison in the context of vehicular embedded systems. In *Proc. of the 31st EUROMICRO conference on Software Engineering and Advanced Applications*, pages 432–441, 2005.

Model-based Toolchain for the Efficient Development of Safety-Relevant Automotive Embedded Systems

2011-01-0056
Published
04/12/2011

Eric Armengaud and Markus Zoier
Virtual Vehicle

Andreas Baumgart
OFFIS e. V.

Matthias Biehl and DeJiu Chen
Royal Institute of Technology

Gerhard Griessnig
AVL List

Christian Hein and Tom Ritter
Fraunhofer FOKUS

Ramin Tavakoli Kolagari
Volvo Technology Corporation

doi: 10.4271/2011-01-0056

ABSTRACT

Advanced functionalities unthinkable a few decades ago are now being introduced into automotive vehicles through embedded systems for reasons like emission control, vehicle connectivity, safety and cooperative behaviors. As the development often involves stakeholders from different engineering disciplines and organizations, the complexity due to shared requirements, interdependencies of data, functions, and resources, as well as tight constraints in regards to timing, safety, and resource efficiency makes the system integration, quality control and assurance, reuse and change management increasingly more difficult. This calls for a more rigorous approach to the development of automotive embedded systems and components. This paper describes the CESAR reference technology platform (RTP) that supports the formalization of various engineering concerns in the development of safety-relevant embedded systems and thereby a model-based integration of various tools and methods to form seamless environments or toolchains for the development of such systems.

INTRODUCTION

Embedded systems are important innovation drivers in the automotive industry. They enable the introduction and improvement of advanced functionalities (e.g., active safety, fuel efficiency support) as well as the replacement of mechanical counterparts while saving weight and costs. At the same time, electronics components present strong requirements with regard to robustness and safety and the resulting system complexity is growing exponentially, which leads to increasing costs and tends to decrease the product quality. As most system development today is distributed across the boundaries of enterprises or engineering teams, it is critical that all system descriptions and information exchanges are precise enough but still with the possibility of protecting the intellectual properties (IP) of concern. For safety critical automotive embedded systems, an emerging standard is the ISO 26262 [12], which provides a reference lifecycle representing the domain consensus on the necessary information and workflow to achieve functional safety of E/E systems. All the aforementioned challenges and constraints for the automotive industry call for a more rigorous approach to the system modeling, design, analysis, verification and validation than is the current state of practice for safety-relevant embedded systems.

The CESAR project [10] has been started in that context to improve the engineering efficiency and effectiveness during the development of safety-critical embedded systems. Next to significantly improvements in tools and methods of the system engineering the R&D project relies on the development of an interoperability platform which is called the CESAR reference technology platform (RTP). The main contributions of this paper are (1) the description of the CESAR RTP concepts, (2) the discussion regarding the tailoring activities required for the deployment of a RTP instance, and (3) the illustration of the benefits from an industrial use case. The use-case is based on EAST-ADL [3], which is an architecture description language allowing the formalization of automotive embedded systems and thereby bringing a potential for a wide range of benefits in regards to system integration, quality control and assurance, and the creation of seamless toolchains for improving the development of safety-relevant automotive embedded systems.

The paper is organized as follows: We first introduce the base technologies and their related state-of-the art approaches in the second section. The base technologies include: (1) the CESAR RTP concept, (2) the interoperability concept for model transformation and seamless tool integration and (3) the CESAR meta-model. Next, in Section 3, we discuss the tailoring activities required for the deployment of a RTP instance. The activities include (1) identification of the development process as well as tools involved, (2) integration of the foreign meta-models of involved tools by means of the CESAR meta-model, (3) development of tool adapters for expected tool interoperations, and (4) integration of the tool adaptation and interoperation components to form a model-based integrated toolchain (tailored RTP instance). In the fourth section, we show how the proposed CESAR RTP facilitates the system design with incremental verification and validation in a component-based development of automotive embedded systems. This is illustrated by an industrial scenario consisting of a system development performed by a dedicated toolchain (RTP instance). The last section concludes this work.

CESAR BASE TECHNOLOGIES AND STATE OF THE ART

THE CESAR RTP CONCEPT

The CESAR project is a European project funded from Artemis JU and national authorities and regroups 55 partners with a global budget of 58 M€ and a cumulated effort of 427 man-years over a duration of three years. CESAR's main objective is the reduction of cost for the development of safety-critical systems by improving the processes and methods for design decisions, analysis and V&V, reuse and change management. The CESAR RTP aims to facilitate the creation of integrated development environments and toolchains for various application domains. Furthermore, the CESAR project also aims to bring significant innovations in the following two fields: (1) requirements engineering, in particular by formalizing multi-viewpoint, multi-criteria, and multi-level requirements, and (2) component-based engineering applied to design space exploration comprising multi-view, multi-criteria and multi-level architecture trade-offs. These innovations are driven - and will be validated - by pilot applications from the domains of automotive, aerospace, rail and automation respectively.

The RTP is a generic model-based integration platform providing a conglomerate of modules that can be combined in different ways in order to provide a seamless development environment (RTP instance) tailored to one specific product development process (see Figure 1). The combinable modules

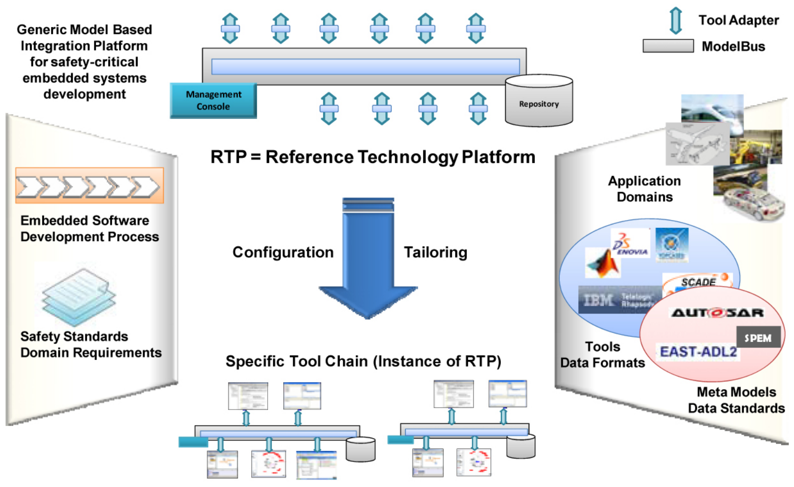

Generic Model Based Integration Platform for safety-critical embedded systems development
Tool Adapter
ModelBus
Management Console
Repository
RTP = Reference Technology Platform
Embedded Software Development Process
Safety Standards Domain Requirements
Configuration
Tailoring
Specific Tool Chain (Instance of RTP)
Application Domains
Tools Data Formats
SCADE
AUTOSAR
SPEM
EAST-ADL2
IBM
Meta Models Data Standards

Figure 1. The CESAR RTP concept

can either be approved commercial-of-the-shelf (COTS) tools, innovative outcomes of research projects, meta models, tool integration frameworks or supporting services. The subset of required entities can be defined using tailoring rules according to relevant safety standards (e.g., ISO 26262, EN 50128), application domain (e.g., automation, aerospace) or company-specific habits (e.g., preferred tools, customized processes). Depending on these rules, a process-driven tailoring of the RTP can be performed. The tailoring step comprises the generation of a customized integrated tooling environment tailored to relevant safety standards, application domain and company habits. This generation comprises activities like tool installation, deployment of adapters for the integration of tool-specific meta-models, and instantiation of a process manager according to company-specific process needs.

The result of the RTP tailoring step is the RTP instance (toolchain). It consists of a set of communicable and interoperable engineering tools as well as the RTP-ModelBus [11] being in charge of the communication between these tools. While each tool is used to support certain design, analysis, and synthesis activities in one or multiple system refinement steps, the ModelBus helps to ensure the data consistency and traceability among different tools. To this end, a dedicated meta-model for this RTP instance is derived from the CESAR meta-model and used to manage interchangeability between the tools by mapping their internal model representations and semantics to a commonly agreed basis. Connections to the RTP-ModelBus are either realized via adapters performing model-transformation (in order to convert tool-internal model representations into the RTP meta-model in case of a tool-specific data format), or directly integrated to the RTP meta-model (when the tool provides a data format compatible with the RTP meta-model). Analysis tools (such as [17]) are connected in the same way.

Existing approaches to interoperability and tool integration for safety-critical embedded systems development are evolving [2, 18] and often academic [7, 8]. However, CESAR aims at achieving a common understanding and an agreement on concepts for interoperability and tool integration for safety-critical embedded systems development between industrial and academic key players in the automotive, aerospace, automation and rail domains. Moreover CESAR aims at a flexible approach where both, solutions of vendors and SMEs as well as open-source solutions can compete in offering best-in-class solutions for particular stages of these processes. This approach is based on cross-domain commonalities in the development processes for safety-critical embedded systems.

In order to gain acceptance of industrial stakeholders and tool vendors, the RTP needs to support integration of existing tool frameworks, often equipped with own meta-models, as well as requirement and configuration management facilities. Thus, the implementation of the RTP instance is strongly driven by the need to identify common aspects in these environments, to provide a consistent view based on them, and to make them available to design, analysis and validation methods. This is facilitated by the development of a proper meta-model for the RTP instance. Thus CESAR fully considers the strong and ongoing trend towards meta-model based tool integration of recent research activities [16].

THE CESAR MODELBUS AND RELATED INTEROPERABILTY CONCEPTS OF INTEGRATED TOOLCHAIN TECHNOLOGIES

In large distributed development environments the consistency of data and processes as well as the synchronization of the communication among the stakeholders is a challenging task. The complexity of efficient tool interaction (orchestration, control, integration) grows with the number of tools involved. The problem domain of tool integration and its theoretical reasons have been identified and discussed from different viewpoints. Wassermann [24] identified five dimensions of tool integration: (1) *Data integration* shares the data produced by different tools and manages the relationship between the data objects. (2) *Control integration* allows tools to notify and activate other tools. (3) *Presentation integration* provides a common user interface with a common look-and-feel. (4) *Process integration* provides process management tools with data from development tools. (5) *Platform integration* provides a virtual operating environment for heterogeneous hardware and software. This has been further elaborated later by Thomas and Nejmeh [22] where they emphasized the role of framework services and integrative environments. In a very similar way, the “Toaster Model” of the European Computer Manufactures Associations (ECMA) [25] addressed this problem domain.

From a practical point of view tool integration is still an open issue. There are a number of vendor specific or point to point integration. Such integration solutions allow the sharing of data among different tools by simple import and export commands using proprietary file formats synchronized via a file system. These solutions work usually only for a limited part of the process and binds the stakeholders to the respective vendor. There are more general approaches to that, often named as Application Lifecycle Management (ALM), but mostly with special emphasis on change management, traceability and reporting. There are communities such as the Open Services for Lifecycle Collaboration initiative, which targets on the definition of vendor-neutral tool interfaces based on Representational State Transfer (REST) for ALM solutions. Apart from ALM, the Eclipse IDE as tool integration platform is in particular strong in the presentation integration aspect (third aspect listed in the previous paragraph). Eclipse IDE is a diverse and flexible solution, which comes with many extensions build by Open Source communities and vendors addressing some of the other integration aspects listed before as well. However, all solutions do have drawbacks, which make it hard to use them in a scope that is addressed by the CESAR project. Therefore, CESAR has defined an interoperability specification, which addresses all relevant aspects of tool integration. This specification defines how tools interact, by defining communication standards and protocols, basic infrastructure services, and data definition and handling. So the CESAR approach is vendor neutral, provides a comprehensive view on the tool integration dimensions and allows the flexible creation of development solutions which take the specifics of the stakeholder's development environment, organizational structures and existing tool landscape into account.

The RTP-ModelBus realizes this interoperability specification. It creates a virtual bus topology connecting tools being part of a certain development environment forming an RTP-instance. This comprises tools of potentially all development phases including process steps as requirements engineering, system architecture design, coding, testing or even reporting. The set of tools used in typical development processes can be quite diverse and may contain COTS tools as well as custom made or in-house proprietary ones. The important point about the interoperability specification and the RTP-ModelBus is that it uses existing methods and approaches in order to avoid building everything from scratch and to combine everything in a coherent way.

RTP-ModelBus is taking a service-oriented approach into account (SOA - define the interfaces in terms of protocols and functionalities, thus enabling loose coupling between the connected tools and services as illustrated in Figure 2). Additionally, only open standards with available implementations are used in ModelBus. Tools connected to the ModelBus provide or consume services that contribute to the system development process. These services can be very different in nature. A service can be a report generator, an analysis tool, or a simulation engine, e.g., executing long running simulations. There are client tools, which are used by humans and which usually offer a graphical user interface and there are server tools, which provide a specific functionality and which are not necessarily equipped with a graphical user interface. Of course these categories are not completely disjunctive since a tool may run in batch or server mode while it provides other parts of its functionality via a graphical user interface.

The problem of data integration is solved by ModelBus in a particular way. ModelBus uses models for exchanging data. So any externalized information (provided or consumed via services) is translated in a common representation framework, which is based on meta-modeling principles. This allows a tool-neutral and technology-agnostic handling of the exchanged data. Of course ModelBus can also handle data, which is not represented as model, such as source code or binary files. ModelBus is using an interaction pattern for realizing the actual exchange of data. This interaction pattern is formally defined and is based on a repository service. With this a tool can easily address an arbitrary model (or any other piece of data) just by using its reference. ModelBus does an automatic lookup and fetches the model from the correct repository. The interaction pattern is outlined in Figure 3 and described also in [11].

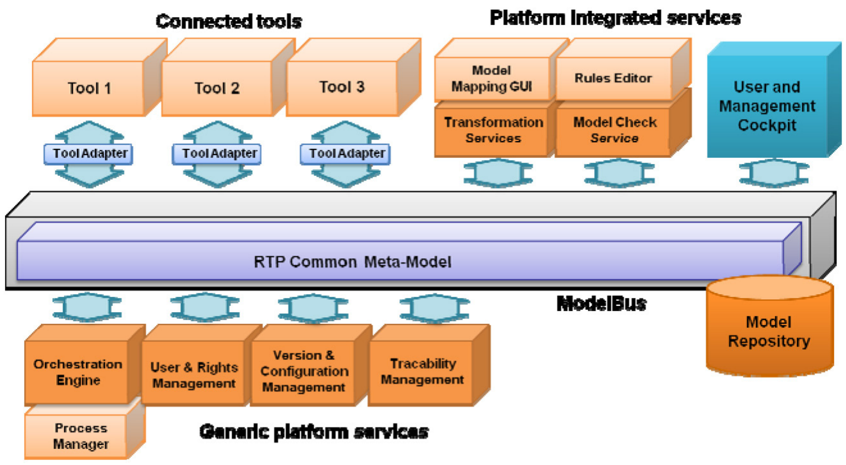


Figure 2. ModelBus overview

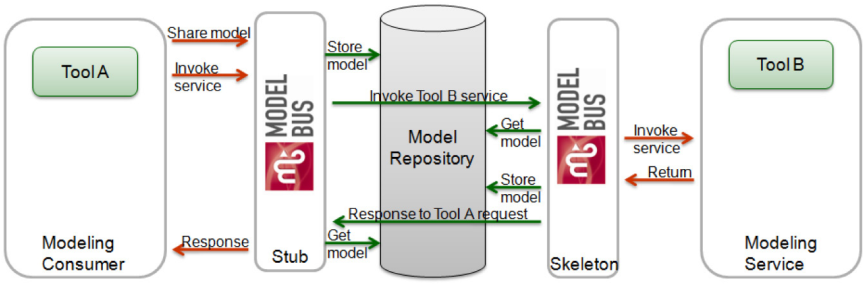


Figure 3. The ModelBus Interaction Pattern

ModelBus is also concerned with the automation of the development process. Certain generic platform services are being used, including an orchestration engine, user & rights management, version & configuration management, and traceability management. The automatic execution of service invocations can be triggered by certain events. Such events can be the provision of a new model or just a new version of a model. ModelBus uses an event distribution service for the realization of the automation. With the help of the automation capability ModelBus offers an additional working mode. A client tool does not necessarily invoke remote services directly, instead it may simply provide its data to the ModelBus and this triggers automatically the invocation of a service. Both working modes (i.e. direct or indirect) can be used in parallel. This helps to create a flexible solution, which fits to the needs of the development process, meets the requirements of the stakeholders, and takes the characteristics of the involved tools into account.

THE CESAR META-MODEL (CMM)

The CESAR project evaluates a common meta-model approach to enable harmonized interoperability between tools and services with heterogeneous data models. It targets to develop a CESAR meta-model (CMM) with common domain-independent meta-model concepts that is modular extendable and which is intended to be used as common meta-model for an RTP instance with an interoperability framework like ModelBus [11]. The CMM therefore provides a harmonized common understanding of data concepts as well as data artifacts and their relationships for the interoperability and integration of tools with heterogeneous data models [17]. In CESAR a multi-domain reference technology platform (RTP) is developed, which can be instantiated for various industrial development processes in safety-critical embedded system design like in avionics, automotive and automation. To enable data interoperability with several integrated tools and services the identification and definition of common meta-model concepts for multiple viewpoints and along all levels of development processes is a crucial point. Based on these common concepts, a domain-, tool- and company-independent and commonly understandable data structure is provided, which can be used for data exchange and analysis increasing tool reuse independent of specific domains and improving integration. The CMM is based on results of European research projects, in which widely usable meta-model concepts were developed, such as EAST-ADL from the ATESST series projects [3] and HRC from the SPEEDS project [20].

The basis of the CMM is an integration of HRC Level 1 Kernel as well as common domain-independent meta-model concepts from EAST-ADL and those developed and identified in CESAR. EAST-ADL complements the HRC/SPEEDS approach with automotive domain-specific concepts regarding architecture, verification and validation, abstraction levels, and lifecycle information. The CESAR meta-model concepts furthermore consider generic component-based design along embedded system development processes with informal as well as formal requirements using different kinds of requirement specification languages (RSL) and traceability management [11].The CMM is therefore intended to support multi-viewpoint component structure modeling along a multi-level development process with requirements, traceability, error modeling as well as verification and validation. It is not a conglomeration of various artifacts to cover everything from all tools and domains because such a meta-model would be giant, unusable and not maintainable [5]. Thus, it provides artifacts with respective relationships to cover common meta-model concepts, which are independent of specific tools and domains.

The current CMM provides core concepts to describe systems along several levels of abstraction by using models with multiple viewpoints on requirements, components as well as V&V cases with respective means of traceability. The CMM enhances HRC components and allows distinction between functional and different kinds of component models. Informal as well as formal requirements based on contracts can be defined and linked to components, requirements and verification and validation entities.

HRC - Heterogeneous Rich Component model

HRC [21] is a meta-model for rich component design supporting modeling and analysis of complex embedded systems by using formal analysis tools. It enables a generic and formal description of components, their decomposition, interfaces and assumed as well as promised behavior in terms of contract-based design. The HRC meta-model provides concepts for component structures, component behavior as well as an expression and action language. Structurally rich components are component types that can be decomposed by defining rich component properties respectively typed by rich components. Rich components have ports, which are typed by port specifications that contain data flows or provide respectively require services which is compliant to ports defined in UML/SysML [14]. Rich component dynamics can be formally described by using state machines and contracts. Contracts are pairs of assumptions and promises. An assumption describes how the component's context is considered to behave and the promise describes how the component behaves if the assumption is fulfilled. HRC state machines allow the description of rich component behavior as hybrid automata [6]. The HRC expression and action language defines datatypes, expressions and actions. HRC data types allow the description of highly complex data structures as they are known from common programming languages like e.g., C. Furthermore, primitive types can be used with additional dimensions and units to describe physical types [5]. HRC expression language allows formal descriptions of data content (i.e., values or conditions) based on simple values, their combination and referenced structural data items, functions and actions. The action language provides a means for statements in terms of imperative programming to describe function bodies, service implementations or actions of state machine transitions.

EAST-ADL

One base technology for the CMM is the EAST-ADL, which aims to provide a well-defined information infrastructure for capturing, evaluating, and managing various engineering concerns across system development stages [3]. The language is aligned with ISO 26262 [12] for functional safety of automotive embedded systems in regards to the work flow and information management. The core of EAST-ADL is its support for system architecture description with models on multiple levels of abstraction according to the needs of separation of concerns in the automotive domain. The topmost system description is performed at Vehicle Level capturing the features of an automotive product family. In EAST-ADL, a vehicle-level feature typically represents an end-to-end system functionality (e.g., braking), while a feature in general refers to a trait or characteristic that a system may or may not have. To satisfy certain requirements, a vehicle-level feature obeys constraints on its behavior (e.g., I/O operations, environmental conditions, and vehicle modes), timing (e.g., end-to-end delay), as well as safety (e.g., the deduced safety goals). By means of feature models, one can specify the allowed variability and inheritance hierarchy, which for example running from vehicle longitudinal control feature to braking and retardation features. The realization of a Vehicle Level feature is supported by logical artifacts, specifying for example the implied functions for sensing, controlling, and actuation in feedback control loops. In EAST-ADL, the design of such logical artifacts is first captured by a Functional Analysis Architecture and thereafter detailed by a Functional Design Architecture at Design Level. It is at the Design Level that the characteristics of system resources as given by the Hardware Design Architecture are taken into consideration. Typical design decisions then include the partitioning and allocation of logical functions on hardware resource. Finally, the Implementation Level specifies the actual software and hardware configuration according to AUTOSAR [4]. EAST-ADL provides dedicated language support for capturing the realization hierarchy from Vehicle Level features to Implementation Level solutions. Similar as in SysML [14], each requirement in EAST-ADL can be traced to its derived subrequirements, implied verification and validation (V&V) cases, system artifacts providing the satisfaction, or detailed information about the implied parameters, states, operations, and vehicle modes. As shown in Figure 4, EAST-ADL allows the elements of a System Model to be associated with a set of models capturing the related requirements, verification and validation cases, environmental concerns, and constraints in regards to behavior, timing (e.g., rate and synchronization constraints), variability, and dependability. For example, a behavior model of EAST-ADL can be used to define the vehicle modes, parameter ranges, operation states as implied in textual requirements, as well as the external descriptions (e.g., in Simulink/Matlab) of computations to be implemented by a system function; a dependability model augments the core system model with the descriptions of hazards, failure modes, and error propagation and the derived safety goals and constraints [19]. Current EAST-ADL support for timing incorporates the results of the ITEA2 project TIMMO [23], which provides a formal description language and methodology for dealing with the timing concerns. The EAST-ADL language can be implemented as a UML profile and thus supported by various UML modeling tools. Further harmonization is being carried out with the aim of releasing the EAST-ADL profile as an annex to the subsequent version of MARTE [13].

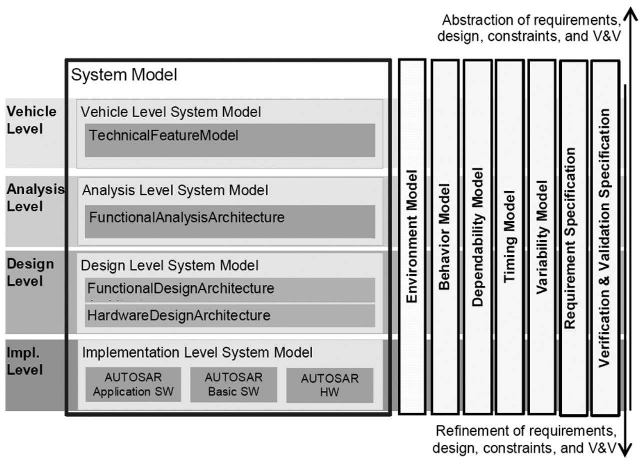

Abstraction of requirements, design, constraints, and V&V
System Model
Vehicle Level
Vehicle Level System Model
TechnicalFeatureModel
Analysis Level
Analysis Level System Model
FunctionalAnalysisArchitecture
Design Level
Design Level System Model
FunctionalDesignArchitecture
HardwareDesignArchitecture
Impl. Level
Implementation Level System Model
AUTOSAR Application SW
AUTOSAR Basic SW
AUTOSAR HW
Environment Model
Behavior Model
Dependability Model
Timing Model
Variability Model
Requirement Specification
Verification & Validation Specification
Refinement of requirements, design, constraints, and V&V

Figure 4. An overview of modeling scope and levels-of-abstraction by EAST-ADL

TAILORING ACTIVITIES FOR THE DEPLOYMENT OF A RTP INSTANCE

OVERVIEW

The tailoring step regroups all the activities required for the instantiation of a dedicated toolchain for a specific purpose, see Figure 1 and Section II-1: CESAR RTP concept. In this Section, we discuss the activities required for the tailoring of the generic assets (e.g., meta-model, tool adapters, data backbone). Figure 5 illustrates the four main tasks required. The first step of the tailoring activities consists of the definition of the development process and the decision of the required development steps and respecting tools. This includes further the deployment of the data backbone (ModelBus) and its configuration according to the tools to be integrated, the design of a workflow as interaction pattern of the platform services (orchestration) and model-transformation services required. The second step concerns the definition of the product meta-model and regroups two sub-activities. The first sub-activity is the identification of the data structure according to the included development steps, the respective viewpoints modeled and the tools involved. This data structure is a union of all the data structures of the tools involved. The second sub-activity is the identification of the required traceability links. These links (especially across the tools) define the relations between the development steps and therefore the required model transformations. The third step consists of the implementation of the specific tool adapters. Such tool adapters provide standard services for the integration to the ModelBus (e.g., check-in, check-out), as well as syntactic and semantic transformation in order to map the tool-specific data to the RTP instance. The last step concerns the evaluation of the integrated toolchain according to typical product development activities.

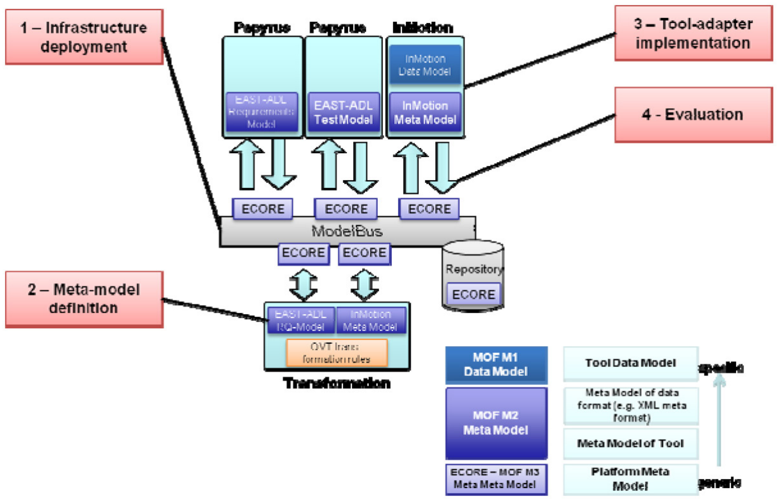


Figure 5. The four different tasks for the development of an RTP instance

META-MODEL INTEGRATION

The CESAR meta-model (CMM) is intended to be used as a shared common model of data in an interoperability platform for tools with heterogeneous data sets. Such data sets have their own types and meta-models such as EAST-ADL. Thus, one crucial issue to enable interoperability among tools in a tailored RTP is the integration of foreign meta-models and to make them compatible with the common meta-meta of the RTP, the CMM. The most important step of such a meta-model integration in a tailoring process is to define a mapping between the foreign meta-model and the CMM. This enables common understanding of the foreign meta-model within the RTP using the CMM. A mapping can be defined in a human readable form like a table. The mapping defines how foreign meta-model concepts with their elements and relationships can be mapped to those of the CMM having common semantics. In addition to the elements and references to other elements a mapping of all information such as the possible number of referenced elements is important for implementation decisions. I.e. for the mapping of the decomposition of EAST-ADL Analysis Function Types the element "AnalysisFunctionTypes", its reference "part", the referenced element "AnalysisFunctionPrototype" and the possible number of referenced Analysis Function Prototypes "0..*" (arbitrarily many referenced elements) have to be listed. Non-trivial mappings regarding combinations of elements and surjective mappings where one element is mapped to one of many elements can be described the following way.

Table 1-1 shows an exemplary mapping between EAST-ADL and the current CMM. This table lists how EAST-ADL and CMM concepts relate to each other and therefore how meta-model elements and relationships can be mapped. For each mapped meta-model concept the respective element name, reference name, referenced element and the possible number of referenced elements are listed. Some mappings are trivial such as RichComponent part and AnalysisFunction part because RichComponent and AnalysisFunction as well as RichComponentProperty and AnalysisFunctionPrototype can be mapped to each other providing decomposition of components. The part property of AnalysisFunctionType has the same multiplicity and the mapped property type is the same as for Rich Component part. When mapping the execution semantics of EAST-ADL function types with run-to-completion semantics being aligned with AUTOSAR to rich component dynamics, this behavior has to be expressed in terms of CMM behavior and contracts, which describe the dynamics of a rich component. In order to make the mapping of the foreign meta-model usable within the RTP an implementation is needed. Concrete implementations depend on the tools being involved and the development process. In the following we describe two reference implementations of meta-model integration along with EAST-ADL and HRC, which were created within the CESAR project and describe harmonized CMM based interoperation between tools with heterogeneous data-models in an instantiated RTP using tool adapters. Both implementations are based on the same mapping, which had to be defined first.

For RTP version 1.0, EAST-ADL and HRC were integrated and therefore mapped to the CMM. This enables harmonized CMM based interoperability between tools and analysis services that use these meta-models like Papyrus with EAST-ADL plugin [15] and an HRC based virtual integration testing service from SPEEDS project [20] in an instantiated RTP, which is exemplarily shown here. Since function types and hardware elements in EAST-ADL provide compositional structures with typed interfaces, respective connections and behavior, these concepts can be mapped to rich components, rich component properties, ports, interconnections and behavior. Furthermore there are special requirements and trace links which can be identified by concepts of the CMM like requirement, satisfy link, derive link etc [5]. EAST-ADL abstraction levels and contained models provide model container and can therefore be mapped to CMM declaration zones that provide generic concepts of abstraction levels and rich component models. Since the CMM widely consists of HRC concepts HRC mapping is mainly trivial. Rich component structures and datatypes can be directly mapped. But HRC contracts belonging to rich components are mapped to CMM rich components satisfying system requirements in CMM.

EAST-ADL is implemented as a UML profile in the Papyrus tool. Thus, for RTP 1.0 a first implementation of EAST-ADL mapping was created by enhancing a CMM UML profile implementation with EAST-ADL stereotypes specializing CMM stereotypes to cover EAST-ADL-specific information related to the common concepts. Stereotype properties were replaced by respective CMM stereotype properties, they were declared "redefining" when associating other specialized stereotypes respectively with different multiplicities or they were declared "subsets" when denoting subsets of CMM stereotype properties. This implementation allowed export of EAST-ADL models in Rhapsody with a tailored EAST-ADL profile to HRC models for formal analysis. A second implementation of EAST-ADL and HRC tailoring to concepts of the CMM was created by performing model transformation between UML with EAST-ADL profile and HRC. Here the identified common concepts of UML with EAST-ADL profile and HRC were used to define QVT transformation rules [5]. Using this technique in a CESAR scenario EAST-ADL structure was enhanced by formal contracts and afterwards formally analyzed by using a virtual integration testing service as defined in SPEEDS project [20].

When an RTP is instantiated a meta-model and an interoperability platform are chosen. The CMM and ModelBus [11] are intended to be used for that purpose. In order to connect tools from a tool chain and services to the interoperability platform in the instantiated RTP tool adapters are chosen. These tool adapter encapsulate an implementation of the mapping between the respective tool data model and the

Table 1-1. Mapping table for EAST-ADL and the CMM

EAST-ADL 2.1				CESAR Meta-Model			
Element	**Reference**	**Referenced Element**	**Number of ref. elements**	**Element**	**Reference**	**Referenced Element**	**Number of ref. elements**
Analysis Function Type	part	Analysis Function Prototype	0..*	Rich Component	part	AnalysisFunction Prototype	0..*
Hardware Component Type	part	Hardware Component Prototype	0..*	Rich Component	part	Hardware Component Prototype	0..*
FunctionType	port	FunctionPort	0..*	Rich Component	port	FunctionPort	0..*
FunctionType	connector	Function Connector	0..*	Rich Component	connector	Function Connector	0..*
Function FlowPort	type	EADatatype	1	Flow in Port typed by Port Specification	type	EADatatype	1
Function ClientServer Port	type	Function ClientServer Interface	1	Port	type	FunctionClient ServerInterface	1
Requirement	text	String	1	System Requirement	text	String	1
Satisfy	satisfied Requirement	Requirement	0..*	Satisfy	satisfied Requirement	Requirement	0..*

meta-model of the RTP instance. In the interoperability platform there can be several data repositories available which are provided by the connected tools or by the interoperability platform itself. On the one hand the CMM or the respective RTP meta-model can be used by the tool adapter as a native data format for data storage in a connected data repository. On the other hand the tool-adapters can store native tool data to an available RTP repository and provide an RTP meta-model view on this data as described in [5]. Using the RTP common meta-model the heterogeneous data of different tools which is available in the RTP can be accessed from all connected tools and data items can be linked to each other in a harmonized data view.

Thus, as a conclusion the integration of foreign meta-models in an RTP tailoring process and the definition as well as the implementation of mappings to the CMM enables common understanding without the concrete knowledge of these meta-models in all tools of the RTP instance. The definition of the mapping between artifacts and relationships of a foreign meta-model and the CMM is very important. Such a mapping can be implemented in several ways to have common understanding of foreign data that is communicated in the RTP instance. In order to enable a harmonized CMM based interoperation between tools with heterogeneous data-models in an instantiated RTP the implementation of a mapping between a tool data-model and the CMM is encapsulated in a tool adapter which can be chosen for the connection of the respective tool to the RTP.

TOOL ADAPTER

In terms of the CESAR approach a tool adapter is needed in order to connect a tool to the RTP-ModelBus. However, there is no universal adapter available that can be used for every tool. An adapter is in most cases specific and it may even be possible that several adapters are needed for the same tool. For example, a text processor tool can be used as requirements specification tool as well as an analysis reporting tool. The relevant data to be exchanged is different in both cases and the adapter can be specific for each purpose. Therefore, an adapter has to take care of the syntax, semantics and operational aspects of the tool and the tool data. In order to implement an adapter a couple of general architectural decisions needs to be taken. At first, it needs to be decided how the tool should interact with others in an integrated toolchain and e.g., whether the tool is used as client or as server tool. A server tool may need to offer specific functionality (e.g., automated execution of a simulation run). In this case a service interface needs to be defined and a server adapter has to be created. If the tool is a client tool (i.e., humans are interacting with it) it has to be decided whether the tool only publishes its internal data to the ModelBus (which may trigger automatic invocation of other services) or whether the tool is about to invoke additionally a remote service. In order to publish the data to ModelBus only, the tool adapter simply needs to use the ModelBus API for storing a model. For invocation of a remote service the adapter has to implement a client for the respective service interface.

Before actually starting the definition and implementation of adapters, another aspect needs to be considered: It has to be decided which data (models) shall be exchanged. Shall a tool export its complete data or shall it export just a subset of it? The CESAR project tries to harmonize the data structures by defining a common meta-model (CMM). So a tool may exports its data adhering to that meta-model, when the adapter is newly created. On the other side existing adapters may export data conforming to other meta-models. In such case a model transformation which translates from the tool specific model into the common meta-model is needed. Such a model transformation can be executed automatically within ModelBus, whenever a new model is exposed.

The following paragraph outlines the general tasks to accomplish in order to create a tool adapter.

Definition of the Service Interface: The easiest way to define a service interface is to use the Java language. For doing so just a Java interface needs to be defined. In this Java interface the concrete model types can be used. By using the ModelBus SDK (based on DOSGI[1]) a web service description language based interface description is derived implicitly when appropriate annotations are being used. The following example shows the definition of a simple service interface containing a service that adds a writer to a library.

The service and the client can be implemented using different technologies as long as the ModelBus interaction pattern is respected. ModelBus is providing skeletons for Java and for . Net. The following code fragment shows the implementation of the service.

The implementation of a data provider for a client tool is very similar. The only difference is that the service interface is already fixed and implemented by the ModelBus repository service. The usage is simple: A session object containing user credentials needs to be created for getting access to the user repository.

```
@WebService(targetNamespace = "http://www.modelbus.org/LibraryService/", name =
"LibraryService")public interface LibraryService {
      @WebMethod(action = "http://www.modelbus.org/LibraryService/addWriter")
      public void addWriter(@WebParam(name = "library", targetNamespace =
        "http://www.modelbus.org/LibraryService/") Library library,
                  @WebParam(name = "name", targetNamespace =
        "http://www.modelbus.org/LibraryService/") String name
      );
 );
```

```
    @Override
    public void addWriter(final Library library, final String name) {
          final Writer writer = LibraryFactory.eINSTANCE.createWriter();
          writer.setName(name);

          library.getWriters().add(writer);
          final Resource res = library.eResource();
    };
```

```
        resource.setURI(URI.createURI("http://mycompany.org/my.uml"));
        […]
        repositoryHelper.checkInModel(session, resource, res.getURI());
```

1 www.dosgi.com

The next step is the integration of the adapters and the respective tools. The realization of adapters can be different and is dependent on the tool. Many COTS tools provide a plug-in mechanism, which allows the integration of ModelBus adapter code in the host tool in an easy way.

INTEGRATION IN MODELBUS AND COMPLETION OF THE TOOLCHAIN

Combining all entities to an integrated development environment takes several steps. The first step is the setup of the generic ModelBus platform. It operates as data backbone connecting all internal services (like transformation services, orchestration, and security) and external services (mainly the tools connected by tool-adapters) necessary for the desired development process. These services and the ModelBus itself are usually deployed on a dedicated server machine running a web application server like Apache Geronimo[2]. In the second step tool adapters are provided to the tools fulfilling the development process. Tool adapters can extend tools by acting as plug-ins or being standalone applications connecting the data management of the tool (usually running on the computers of the development engineers) with the ModelBus platform using a local computer network.

The third step is the configuration of the platform services. Providing the necessary transformation rules (e.g., modeled in QVT[3] or ATL[4]) to the transformation services is the most important step to enable the model-based data exchange. This also includes the configuration of the traceability management. To automate certain parts of the development process and platform background activities, a workflow is designed by a BPMN orchestration scheme[5] as interaction pattern of the platform services. This orchestration scheme is a breakdown of the development work flow to the level of the services and data elements. Now the toolchain setup is completed. To evaluate the tool interaction, reference data (examples of data from development process) is provided by executing the desired development process.

Comparing the provided integrated solution to a straight point-to-point integration of the tools using connectors it is obvious that the direct connection of tools leads to conversion instructions that are separated over several connectors and do not follow an overall pattern. The information exchange between different development activities across development phases is difficult to implement. The communication to activities linking to multiple steps of the development process like requirements- or workflow management is difficult to implement. If an engineering department is running multiple projects at the same time, direct connections get inflexible and are difficult to handle. An integrated solution using a common, model based scheme allows a concentration of all conversation instructions in the platform following one generic concept (meta-model of the RTP instance) and using standardized interfaces.

USE-CASE: AUTOMOTIVE TEST MANAGEMENT TOOLCHAIN

A demonstrator has been implemented in order to illustrate our claims, see Figure 6. The motivation for the integrated toolchain is to support the interactions between the roles of requirements engineer, system engineer, V&V manager and test engineer. The toolchain includes the connection of the tools Papyrus[6] and the AVL InMotion[7] using the RTP-ModelBus presented in Section 2. Papyrus, on one side, is an open-source tool that supports the modeling of automotive embedded systems according to EAST-ADL. AVL InMotion, on the other side, is a real-time simulation platform for maneuver and event-based testing at the test bed. In combination with AVL Cruise[5], it supports key business objectives such as hybridization and electrification of power train engineering.

For the evaluation, a recuperation function for a hybrid vehicle has been used. Recuperation is the recovery of kinetic energy by the e-motor operating in generator mode. Here the e-motor generates a negative torque that decelerates the vehicle and is transformed into electrical energy that is used to charge the batteries and to supply the conventional low voltage board net with electrical energy. The system regroups 28 requirements and can be decomposed into five sub-components. Two test campaigns have been developed to test the requirements. Seven test cases have been automatically derived from these two test campaigns.

Figure 7 provides an overview of the resulting model and the mapping with the development activities. During the first step, the system requirements are modeled (RECUP) and refined (RECUP1 and RECUP2). During the second step, the system architecture is defined (Hybrid Control Unit, E-Drive) and the traceability links are added. The third step consists of the description of the test cases (TestCase RECUP1 Motoring, TestCase RECUP2 Recuperation). It includes links to a reference test case consisting of the description of the car, its behavior and the environment as well as variation parameters for the car behavior (e.g., car acceleration or pressure on the

2 http://geronimo.apache.org/
3 QVT = Query/View/Transformation, http://www.omg.org/spec/QVT/
4 ATL = Atlas model transformation language, http://www.eclipse.org/atl/
5 www.bpmn.org
6 papyrus-uml.org
7 www.avl.com, AVL InMotion powered by IPG CarMaker

brake pedal). These three first steps are performed in the Papyrus tool according to the EAST-ADL methodology. The fourth step is the model transformation from EAST-ADL to the AVL InMotion tool. The variation parameters described in step three are the inputs to generate automatically an InMotion test case for each parameter variant. In the fifth step, the test cases are implemented in the corresponding test environment and executed. The results are then transformed back to the Papyrus tool (step six). During this InMotion to EAST-ADL transformation, the test execution status and a link to the test results from InMotion are automatically inserted into the EAST-ADL model. Finally the results are available for analysis for the V&V manager (step seven).

The main benefits of the proposed integrated tool-chain for the recuperation function are the following:

• Explicit annotation of the system architecture using a semi-formal language (EAST-ADL). This minimizes the ambiguity of the system being developed and simplifies synchronization between large teams (multi-site collaboration). Furthermore, the formalization provides a solid basis for further static analysis (e.g., completeness checks: are all the components mapped to requirements and test cases?). The modeling of the system using EAST-ADL has enabled a fast synchronization of our 15 person team across three countries.

• Traceability between requirements, system components, test cases and test results. Thanks to the meta-model covering the entire development process, all the activities are related together and the collaboration between the different experts is strongly improved. During our evaluation, the misunderstandings between the different experts and work steps have been strongly reduced thanks to the complete overview of the development status.

• Easy extension of the current toolchain. The availability of a vendor-neutral data backbone with standardized services as well as common understanding on the data thanks to the CESAR meta-model makes the enhancement of the toolchain quite easy. Hence, further tools (e.g., for system design, analysis or validation) can be integrated on the platform later on.

• Automated notification when the system has been modified and a new task should be performed. This supports the coordination within the team.

• Automated configuration of following development steps. Consecutive development steps usually represent a refinement of the abstract model down to a specific implementation and its corresponding validation. Results of a development steps can be used as input for the following development step. When different tools are used, the configuration has to be performed manually, which is error prone. Model transformation with tool adapters enables the automation of data transfer, thus reducing development time and minimizing human error. During our evaluation, model transformation has been used (1) for automated test case generation out of the test campaigns and (2) for mapping the test results back to the system description. These transformations have saved engineering time and have minimized human error.

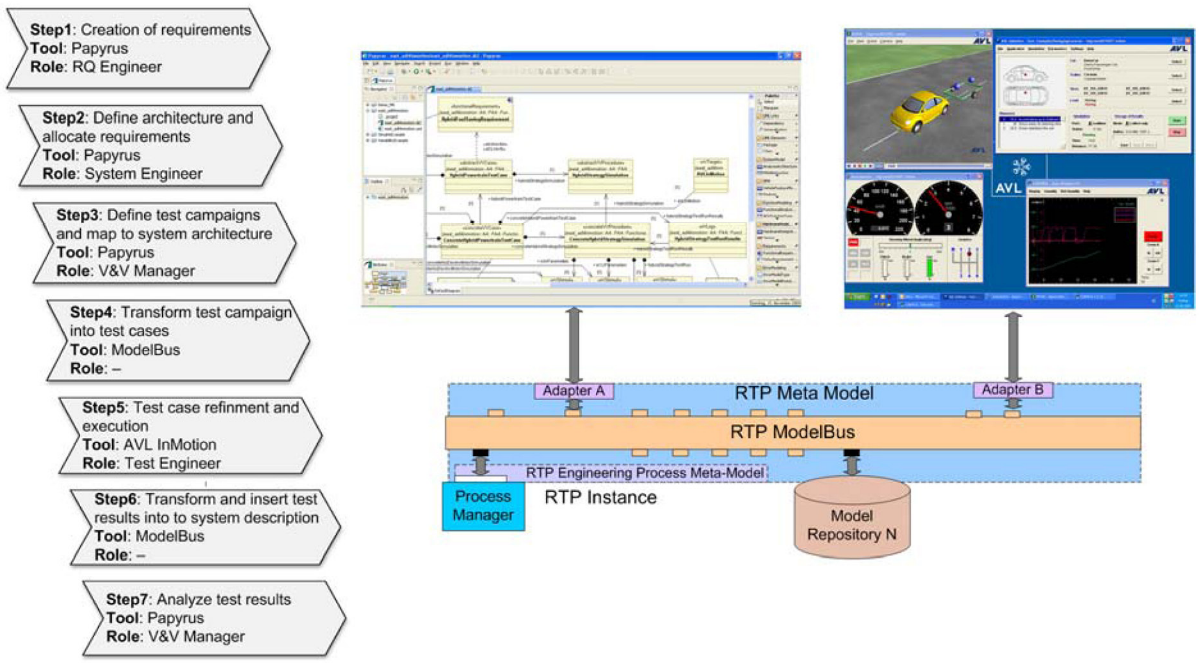

Step1: Creation of requirements
Tool: Papyrus
Role: RQ Engineer
Step2: Define architecture and allocate requirements
Tool: Papyrus
Role: System Engineer
Step3: Define test campaigns and map to system architecture
Tool: Papyrus
Role: V&V Manager
Step4: Transform test campaign into test cases
Tool: ModelBus
Role: –
Step5: Test case refinment and execution
Tool: AVL InMotion
Role: Test Engineer
Step6: Transform and insert test results into to system description
Tool: ModelBus
Role: –
Step7: Analyze test results
Tool: Papyrus
Role: V&V Manager
Adapter A
RTP Meta Model
Adapter B
RTP ModelBus
RTP Engineering Process Meta-Model
Process Manager
RTP Instance
Model Repository N
AVL

Figure 6. RTP instance for automotive test management scenario

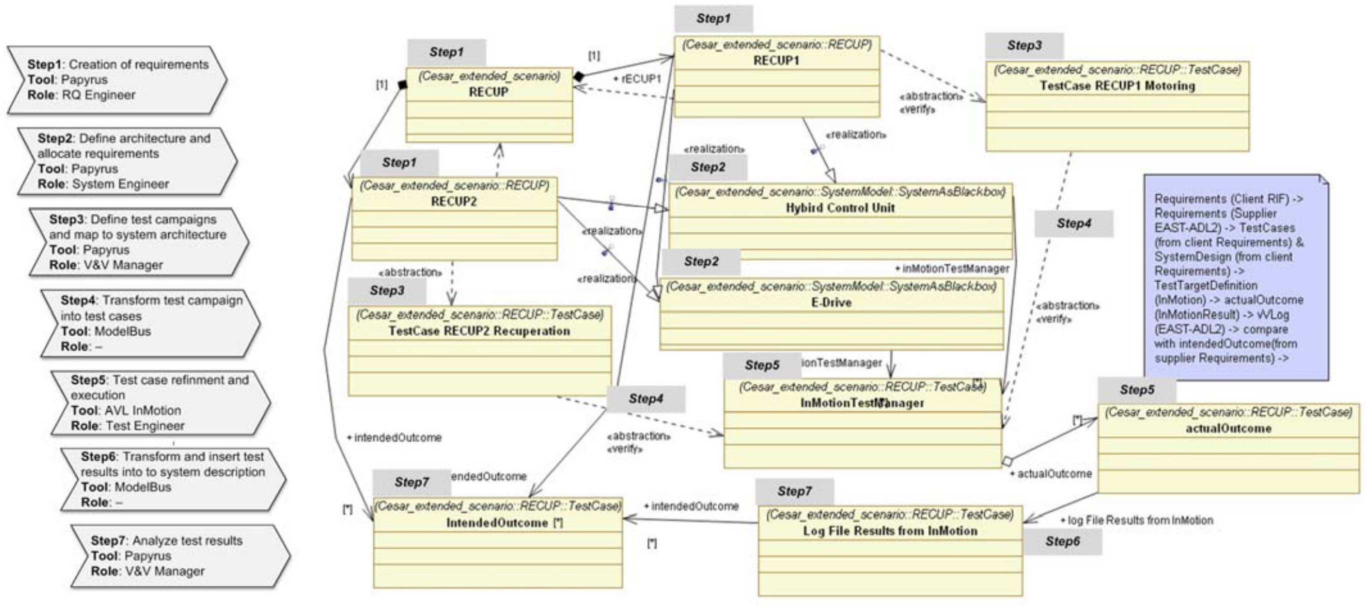

Step1: Creation of requirements
Tool: Papyrus
Role: RQ Engineer
Step2: Define architecture and allocate requirements
Tool: Papyrus
Role: System Engineer
Step3: Define test campaigns and map to system architecture
Tool: Papyrus
Role: V&V Manager
Step4: Transform test campaign into test cases
Tool: ModelBus
Role: –
Step5: Test case refinment and execution
Tool: AVL InMotion
Role: Test Engineer
Step6: Transform and insert test results into to system description
Tool: ModelBus
Role: –
Step7: Analyze test results
Tool: Papyrus
Role: V&V Manager
(Cesar_extended_scenario) RECUP
(Cesar_extended_scenario::RECUP) RECUP1
(Cesar_extended_scenario::RECUP) RECUP2
(Cesar_extended_scenario::RECUP::TestCase) TestCase RECUP1 Motoring
(Cesar_extended_scenario::SystemModel::SystemAsBlackbox) Hybird Control Unit
(Cesar_extended_scenario::SystemModel::SystemAsBlackbox) E-Drive
(Cesar_extended_scenario::RECUP::TestCase) TestCase RECUP2 Recuperation
(Cesar_extended_scenario::RECUP::TestCase) InMotionTestManager
(Cesar_extended_scenario::RECUP::TestCase) actualOutcome
(Cesar_extended_scenario::RECUP::TestCase) IntendedOutcome [*]
(Cesar_extended_scenario::RECUP::TestCase) Log File Results from InMotion
«abstraction» «verify»
«realization»
+ rECUP1
+ inMotionTestManager
+ intendedOutcome
+ actualOutcome
+ log File Results from InMotion
Requirements (Client RIF) -> Requirements (Supplier EAST-ADL2) -> TestCases (from client Requirements) & SystemDesign (from client Requirements) -> TestTargetDefinition (InMotion) -> actualOutcome (InMotionResult) -> VVLog (EAST-ADL2) -> compare with intendedOutcome(from supplier Requirements) ->

Figure 7. Model overview of the product development

SUMMARY/CONCLUSIONS

Due to its size and relevance of the partners, the CESAR project provides a platform to enable different technologies - coming both from industry and research projects - to converge. The focus is set to the cost-efficient development of safety-relevant embedded systems. During this work, the CESAR RTP as central concept of the project has been discussed. Deployment of a dedicated RTP instance regroups four tasks which are: 1. identification of the development process as well as tools involved, 2. integration of foreign meta-models to be mapped to the CESAR meta-model, 3. development of tool adapters, and 4. integration of the components into a model-based integrated toolchain (tailoring RTP instance). These four tasks are difficult to coordinate since they regroup different expertise ranging from domain-independent software engineering up to domain-(tool-)specific meta-models. The resulting toolchain, however, strongly improve product development from the following point of views: 1. traceability between the system artifacts (e.g., between requirements, components, test cases) and therefore improvement of product quality, 2. improved tool integration and therefore better cooperation between development steps requiring different tools and expertise, thus saving development time, and 3. advanced integration platform supporting the integration of automated services (such as static analysis or automated documentation generation).

REFERENCES

1. AADL - Architecture Analysis & Design Language, http://www.aadl.info

2. Altheide, F., Dörfel, S., Dörr, H., Kanzleiter, J.: An architecture for a sustainable tool integration. In: Proc. of the Workshop on Tool Integration in System Development, European Software Engineering Conference (TIS 2003). (2003) 29-32

3. The ATESST2 Consortium. *EAST-ADL Domain Model Specification*. Advancing Traffic Efficiency and Safety through Software Technology (ATESST). EUROPEAN COMMISSION FP7 Grant Agreement 224442. 2010. <www.atesst.org>

4. AUTOSAR Development Partnership, http://www.autosar.org

5. Baumgart, A. A common meta-model for the interoperation of tools with heterogeneous data models, Proceedings of the 3rd Workshop on Model-Driven Tool & Process Integration (MDTPI, 2010):

6. Baumgart, A.; Reinkemeier, P.; Rettberg, A.; Stierand, I.; Thaden, E.; Weber, R.: A Model-Based Design Methodology with Contracts to Enhance the Development Process of Safety-Critical Systems : Proceedings of 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS, 2010)

7. Burmester, S., Giese, H., Niere, J., Tichy, M., Wadsack, J., Wagner, R., Wendehals, L., Zündorf, A.: Tool integration at the meta-model level: the fujaba approach. International Journal on Software Tools for Technology Transfer (STTT) 6 (2004) 203-218

8. Burmester, S., Giese, H., Hirsch, M., Schilling, D., Tichy, M.: The fujaba real-time tool suite. In: Proc. of the 27th International Conference on Software Engineering (ICSE 2005). (2005) 670-671

9. Earl, A.: Principles of a reference model for computer aided software engineering environ-ments. In Ling, F, editor, The international Workshop on Environments (Software Engineering Environments), volume 647 on Lecture Notes in Computer Sciences, pages 115-129, Springer-Verlag, Berlin, September 1989, Chinon, France

10. Griessnig, G., Mader, R., Peikenkamp, T., Josko, B., Törngren, M., Armengaud, E.: CESAR: Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems, in Embedded World 2010 - ARTEMIS Session

11. Hein, C., Ritter, T., Wagner, M.: Model-driven tool integration with modelbus. In: Proc. of the Workshop Future Trends of Model-Driven Development. (2009)

12. International Organization for Standardization: ISO/DIS 26262 on Functional Safety for Road Vehicles. 2009.

13. OMG: The UML Profile for MARTE - Modeling and Analysis of Real-Time and Embedded Systems. MARTE specification version 1.0 (formal/2009-11-02). http://www.omgmarte.org/

14. OMG: OMG Systems Modeling Language - SysML, V1.2., OMG Document Number: formal/2010-06-01. http://www.sysml.org

15. Papyrus UML, Open Source Tool for Graphical UML2 Modelling, http://www.papyrusuml.org

16. Passerone, R., Ben Hafaiedh, I., Graf, S., Benveniste, A., Cancila, D., Cuccuru, A., Girard, S., Terrier, F., Damm, W., Ferrari, A., Mangeruca, L., Josko, B., Peikenkamp, T., Sangiovanni-Vincentelli, A.: Metamodels in Europe: Languages, tools, and applications. IEEE Design and Test of Computers 26(3) (2009) 38-53

17. Peikenkamp, T., Cavallo, A., Valacca, L., Böde, E., Pretzer, M., Hahn, E.M.: Towards a Unified Model-Based Safety Assessment. In: SAFECOMP. (2006) 275-288

18. Ridderhof, W., Gross, H.G., Dörr, H.: Establishing evidence for safety cases in automotive systems - a case study. In: Proc. of the 26th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2007). (2007) 1-13

19. Sandberg, A., Chen, D., Lönn, H., Johansson, R., Feng, L., Törngren, M., Torchiaro, S., Tavakoli-Kolagari, R., Abele, A.: Model-based Safety Engineering of Interdependent Functions in Automotive Vehicles Using EAST-ADL2. Lecture Notes in Computer Science, 2011, Volume 6351, Computer Safety, Reliability, and Security. Springer (SAFECOMP2010). (2011) 332-346

20. SPEEDS Consortium, SPEEDS (SPECulative and Exploratory Design in System Engineering), European funded project, 2008, http://www.speeds.eu.com/

21. SPEEDS Project: D.2.1.5 SPEEDS L-1 Meta-Model: Deliverable: Rev. 1.0.1: May 2009

22. Thomas, I., Nejmeh, B.: Definitions of Tool Integration for Environments. IEEE Software, 9(2):29-35, March 1992

23. TIMMO - TIMing MOdel. ITEA 2 project 06005. http://www.timmo.org/

24. Wassermann, A.: Tool Integration in software engineering environments. In The International Workshop on Environments (Software Engineering Environments), volume 647 of Lecture Notes in Computer Sciences, pages 137-149, Springer-Verlag, Berlin, September 1989, Chinon, France

CONTACT INFORMATION (ALPHABETICAL ORDER)

Eric Armengaud
Markus Zoier
Virtual Vehicle Competence Center
Inffeldgasse 21a, 8010 Graz, Austria
eric.armengaud@v2c2.at
markus.zoier@v2c2.at

Andreas Baumgart
OFFIS e.V.
Escherweg 2, 26121 Oldenburg, Germany
baumgart@offis.de

Matthias Biehl
Dejiu Chen
KTH Royal Institute of Technology
Valhallavägen 83, 10044 Stockholm, Sweden
chen@md.kth.se
biehl@md.kth.se

Gerhard Griessnig
AVL List GmbH
Hans-List-Platz 1, 8020 Graz, Austria
Institute for Technical Informatics, Graz University of Technology, Austria
gerhard.griessnig@avl.com

Christian Hein
Tom Ritter
Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
christian.hein@fokus.fraunhofer.de
tom.ritter@fokus.fraunhofer.de

Ramin Tavakoli Kolagari
Volvo Technology Corporation
Regnbågsgatan 1, SE-405 08 Gothenburg, Sweden
ramin.tavakoli@volvo.com

Fault Insertion Testing of a Novel CPLD-based Fail-Safe System

Gerhard Grießnig
AVL LIST GMBH
Graz, Austria
gerhard.griessnig@avl.com

Roland Mader, Christian Steger, Reinhold Weiß
Graz University of Technology
Institute for Technical Informatics (ITI)
Graz, Austria
rmader@sbox.tugraz.at, steger@tugraz.at,
rweiss@tugraz.at

Abstract—**According to the standard IEC 61508 fault insertion testing is required for the verification of fail-safe systems. Usually these systems are realized with microcontrollers. Fail-safe systems based on a novel CPLD-based architecture require a different method to perform fault insertion testing than microcontroller-based systems. This paper describes a method to accomplish fault insertion testing of a system based on the novel CPLD-based architecture using the original system hardware. The goal is to verify the realized safety integrity measures of the system by inserting faults and observing the behavior of the system. The described method exploits the fact, that the system contains two channels, where both channels contain a CPLD. During a test one CPLD is configured using a modified programming file. This file is available after the compilation of a VHDL-description, which was modified using saboteurs or mutants. This allows injecting a fault into this CPLD. The other CPLD is configured as fault-free device. The entire system has to detect the injected fault using its safety integrity measures. Consequently it has to enter and/or maintain a safe state.**

Keywords-IEC 61508; fail-safe system; safety integrity; fault insertion testing; fault injection; CPLD; VHDL

I. INTRODUCTION

A new architecture for fail-safe systems [10] has been developed recently. The architecture is homogenously redundant and consists of two channels. In contrast to other fail-safe system architectures [1] this novel architecture contains no microcontrollers. Instead the safety functions and the safety integrity measures are exclusively implemented using one CPLD (complex programmable logic device) per channel.

A fail-safe system based on this novel CPLD-based architecture is being developed by SIEMENS. The application domain of this system is industrial automation. The fail-safe system is able to execute safety functions if demanded. Each channel is able to perform the safety functions independently. Additionally each channel realizes measures to detect faults. This guarantees sufficiently high safety integrity. The functionality of the system is entirely implemented in hardware.

The certification of the fail-safe system in adherence with the standard IEC 61508 [6] is aspired. Thus it is necessary to fulfill the requirements of this standard. One of the verification measures, which are required by the IEC 61508, is fault insertion testing. The accomplishment of fault insertion tests is necessary to verify the realized safety integrity measures of the fail-safe system. This paper presents the new approach of fault insertion testing of a novel CPLD-based fail-safe system.

The rest of this paper is organized as follows. Section II describes the state of the art, reviews related work and introduces the standard IEC 61508, which requires fault insertion testing for safety related systems. In Section III the CPLD-based fail-safe system and its novel architecture are briefly described. The proposed approach to inject faults using a modified VHDL-description and a manual switch is described in Section IV. The procedure to verify the safety integrity measures of the fail-safe system is described in Section V. Section VI describes how synthesis results can be influenced using VHDL-constants. Section VII presents experimental results. Finally Section VIII concludes.

II. STATE OF THE ART

In the domain of industrial automation fault insertion testing is usually applied for the verification of microcontroller-based fail-safe systems. A fault can be injected into a microcontroller by modifying the executed program, which is usually implemented using C or C++. In this case the program can be modified with preprocessor commands, to inject a fault. Then the program needs to be recompiled and loaded into the microcontroller, which has to emulate the fault.

In contrast to microcontroller-based fail-safe systems the fail-safe system based on the novel CPLD-based architecture requires a new method to perform fault insertion testing, because these devices do not execute programs implemented in C or C++. To develop the presented method we reviewed related work and examined the relevant requirements of the standard IEC 61508.

A. Related Work

In [7] VHDL-based fault injection techniques are discussed. A saboteur is a component that is added into a VHDL-description to alter the timing characteristic or the value of a signal, if the saboteur is activated. During the normal operation

978-3-9810801-5-5/DATE09 © 2009 EDAA

of the system the saboteur is inactive. Saboteurs can be used to inject various fault types like stuck-at faults, bit-flips, bridging-faults or delay faults.

Mutants are components which replace corresponding components. If a mutant is activated, it behaves like the corresponding component in presence of a fault. If the mutant is inactive, it behaves like the fault-free corresponding component. A mutant can either be created by adding saboteurs to a structural model description, by replacing subcomponents of a structural model description or by modifying the syntactical structures of a behavioral description [8]. It is possible to inject various fault types like assignment control, stuck-else or stuck-then using mutants [7].

Fault simulations [2] can be used to analyze the behavior of fault tolerant circuits in presence of faults. In [3] the use of PLDs is proposed to accelerate a fault simulation. A PLD is connected to a host computer. The circuit, which has to be simulated, is entirely or partly mapped on the PLD. The host computer executes a simulation program, which applies input vectors to the circuit. The responses are read back by the host computer.

The authors distinguish between dynamic and static fault injection. In case of dynamic fault injection faults are injected during run-time. Multiplexers are inserted into the circuit to emulate stuck-at faults. The multiplexers contain two inputs. One input is connected to the correct input value and the other input is connected to a 1 or a 0 (stuck-at fault). The host computer controls the selector wire of each multiplexer. Thus it can determine if the correct input value or the stuck-at fault is switched to the output of the multiplexer. The advantage of this approach is that the circuit description has to be compiled only once. Also the PLD has to be configured only once. The disadvantage is that the multiplexers require additional PLD resources. Also the delay of the circuit increases, limiting the maximal clock speed.

An alternative approach is static fault injection. In this case faults are injected into the circuit during compile-time. Thus whenever the circuit has to be simulated using another set of faults, the circuit description needs to be recompiled and the PLD needs to be reconfigured. The disadvantage is that the frequent recompilation and reconfiguration increases the duration of the simulation. The advantage is that no additional multiplexers are inserted. Thus no additional PLD resources are needed and the delay of the circuit is not increased.

In [4] the use of various kinds of fault injection elements is proposed to inject faults. These elements contain either one or two test inputs. Depending on the test inputs either correct inputs or stuck-at-faults are switched to the outputs of the fault injection elements. The test inputs of the fault injection elements can be connected to the flip-flops of a fault injection scan chain or to the outputs of a decoder to inject faults.

In [5] the use of a single FPGA as fast simulation environment is described. A faulty version of a circuit and a fault-free version of the same circuit are emulated concurrently. A comparator compares the outputs of the faulty circuit to the outputs of the fault-free circuit. If the outputs differ, a fault has been detected. A LFSR (linear feedback shift register) is used to generate test vectors for the circuits. An additional state machine controls the fault simulation.

B. *IEC 61508*

The IEC 61508 [6] is a basic functional safety standard, which defines a safety life cycle. All activities from the initial concept, through specification, design, implementation, operation to the disposal of the system are covered. The standard IEC 61508 applies, when an E/E/PES (electrical/electronic/programmable electronic system) carries out safety functions.

This standard defines four safety integrity levels (SIL). While SIL 4 is the highest achievable level of safety integrity, SIL 1 is the lowest achievable level. The higher the safety integrity level of a system, the higher the probability that an E/E/PES performs its safety functions correctly if demanded.

Requirements for preventing failures during the development process and requirements for controlling failures, if they are present, are defined by this standard. Techniques and measures, which are necessary to reach a certain level of safety integrity, are specified.

The IEC 61508 requires fault insertion testing for the verification of safety related systems. Thus fault insertion testing is an important prerequisite for the successful certification of a system in adherence with the IEC 61508. The aim of fault insertion tests is to simulate faults in the system hardware [6]. Then the responses of the faulty system are analyzed. These kind of tests are used to assess the dependability of the system in case of a fault.

III. The Novel CPLD-based Fail-Safe System

The application domain of the presented fail-safe system is industrial automation. Assume an electric motor in a manufacturing plant, which is controlled by a PDS (power drive system). This PDS realizes a number of different standard functions, which should be able to stop the electric motor, if demanded. To reduce the risk of failing standard functions, corresponding safety functions have to be implemented. A safety function is able to stop the electric motor, even if the corresponding standard function fails.

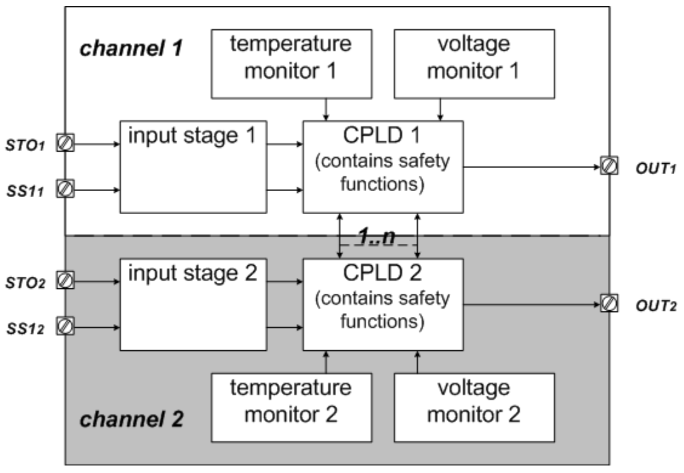


Figure 1. Architecture of the F-module

A. *Architecture of the Fail-Safe System*

The safety functions are realized using a separate module (F-module), which can be connected to the PDS. To communicate with the PDS, the CPLDs exchange messages with a DSP (digital signal processor), which is a component of the PDS. The architecture of the F-module is illustrated by Figure 1 schematically.

The architecture consists of two channels. Each channel is able to perform the safety functions independently. Thus the safety functions can still be carried out, if one channel is faulty. The F-module consists of the following components:

- **Input stages:** The input stages of the F-module are used to activate the safety functions. Each channel contains a separate input stage.
- **CPLDs:** The CPLDs carry out the safety functions, if demanded. Additionally the CPLDs realize measures to increase the safety integrity of the system. Thus they exchange a number of corresponding signal pairs. The CPLDs can deactivate the electric motor via the safety critical outputs (OUT_1 and OUT_2).
- **Voltage monitors:** There are separate voltage monitors, which are able to detect undervoltage and overvoltage. Additionally each channel is protected against dangerous overvoltage.
- **Temperature monitors:** One temperature monitor per channel checks, if the temperature is within the specified range.

B. *Behavior of the F-module*

The behavior of the F-module is illustrated by Figure 2. The system can enter a safe state and an unsafe state. The safe state can be divided into the substates start-up, hard-error and motor stopped. When the system is in the safe state, no power is applied to the electric motor. Thus the motor is not able to cause harm to people, the environment or property. When the system is in state motor running (unsafe), power can be applied and the PDS is able to control the motor.

When the F-module is switched on, the system enters state start-up (safe). In this state the CPLDs initialize all their registers and flip-flops. Both CPLDs receive parameters for a safety function from the DSP. Before the system leaves this state, the CPLDs have to perform an initial test, which ensures that the CPLDs are able to perform the safety functions correctly.

When the initial test is finished successfully by both CPLDs, they tell the PDS via the DSP, that state start-up can be left. With a defined DSP message the system enters state motor running (unsafe).

If a safety function is activated, the system enters state motor stopped (safe). If the safety functions have been performed properly and if they are not activated any more, the DSP can tell the F-module to enter state motor running again.

During the operation of the F-module various safety integrity measures are performed to detect faults. If a fault is detected, the system enters state hard-error (safe). This state cannot be left any more as long as the system is operating.

C. *Safety Integrity Measures*

To guarantee sufficiently high safety integrity, both CPLDs perform diagnostic checks to detect faults as long as the F-module is switched on. If a safety integrity measure detects a fault, the system enters the safe state. Various safety integrity measures to detect faults are realized:

- **Discrepancy monitors:** The CPLDs exchange corresponding signal pairs. Some of them contain information about the safety functions. If these signal pairs are discrepant for a specified time, a fault is signaled.
- **Init-test:** When the system is in state start-up, a test is performed, which verifies that both CPLDs are able to perform the safety functions correctly.
- **SS1-test:** When the Init-test is finished, the CPLDs perform a periodic background test. Components, which are relevant for the correct execution of the safety functions, are tested.
- **Shut down test:** The CPLDs get information about the state of the electric motor via feedback signals. If the deactivation of the motor does not occur in a specified time, a fault is signaled.
- **Short circuit test:** This test can detect short circuits between the safety critical outputs of the F-module.
- **Temperature monitor:** If the temperature of the environment exceeds a certain limit, the system enters the safe state.
- **Voltage monitor:** If the supply voltage of the CPLDs is too high or too low, the system enters the safe state.

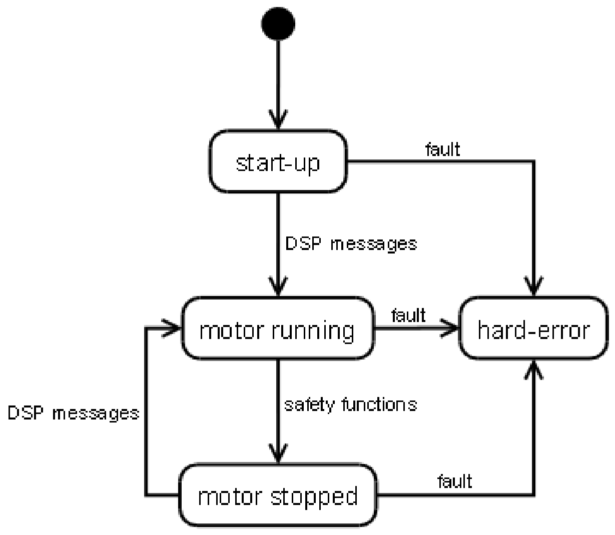


Figure 2. Behavior of the F-module

IV. FAULT INSERTION USING SYSTEM HARDWARE

The IEC 61508 requires fault insertion tests to be performed using the system hardware. That is why it is not sufficient to perform fault insertion tests using software simulations. Instead we use the system hardware (F-module) to perform the tests.

The behavior of the CPLDs on the F-module is defined by a synthesizable VHDL-description. After the compilation of the VHDL-description a programming file is available, which can be used to configure each CPLD via a JTAG interface. Usually both CPLDs are configured using the same programming file. Nevertheless it is possible to configure the CPLDs using different programming files. The presented method to perform fault insertion testing exploits this fact.

To inject a fault we create a modified version of the synthesizable VHDL-description. This modified description makes it possible to inject a certain permanent or transient fault into one CPLD via an input pin of this CPLD. After compilation a modified programming file is available. We configure one CPLD (fault emulating device) with this modified programming file. The other CPLD (golden device) is configured with the original programming file. Thus it is possible to inject a fault into one CPLD but not into the other one.

One input pin of the fault emulating device is connected to a manual fault enable switch. When the input pin is not switched to V_{DD}, no fault is injected and the fault emulating device has the same behavior as the golden device. If the input pin is switched to V_{DD}, a fault is injected into the fault emulating device.

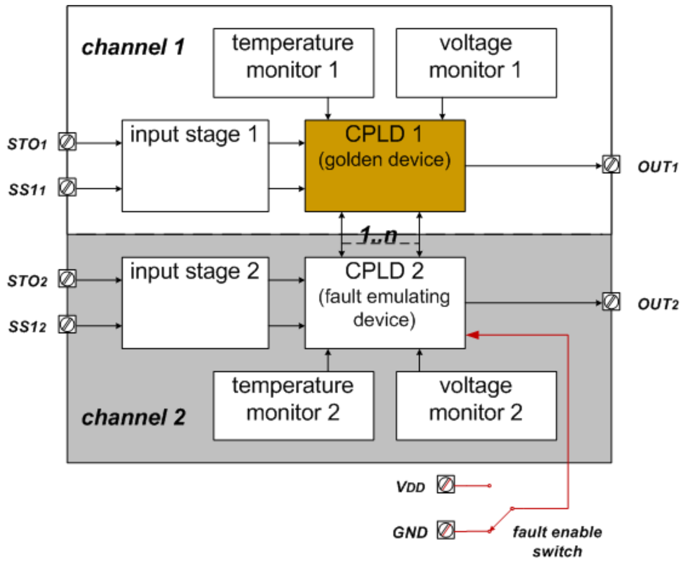


Figure 3. Fault insertion using the F-module

To modify the VHDL-description saboteurs can be added. This allows the injection of transient faults like bit-flips. Also permanent and intermittent faults like stuck-at faults, bridging-faults and delay faults can be injected. Also mutants can be added to the VHDL-description to inject faults. Saboteurs and mutants can be activated and deactivated via the fault enable switch. Various fault models can be considered using the presented approach. The only restriction for the creation of saboteurs and mutants for this approach is the use of synthesizable VHDL-constructs.

Figure 3 illustrates the F-module, which is prepared for fault insertion testing. In this case CPLD 1 is configured as golden device and CPLD 2 is configured as fault emulating device. Generally it does not matter which CPLD is configured as golden device and which CPLD is configured as fault emulating device.

V. VERIFICATION OF SAFETY INTEGRITY MEASURES

According to the requirements of IEC 61508 the synthesizable VHDL-description is modular and consists of a number of interconnected components. We perform a system FMEA (failure modes and effects analysis) and a design FMEA to determine possible sources of failure of the components and their interconnections and identify the consequences in terms of system behavior.

If a fault of a component or interconnection leads to the safe state of the entire system, the fault can be considered to be safe. In this case no fault insertion test case needs to be planned. If a fault potentially leads to the unsafe sate of the entire system, we expect the implemented safety integrity measures to detect the fault and the system has to enter and/or maintain the safe sate. Consequently a test case needs to be planned to verify that the safety integrity measures are able to detect the fault.

It is necessary to verify, that each of the two channels is able to detect injected faults. Thus CPLD 1 is configured as golden device for one half of the test cases, while CPLD 2 is configured as golden device for the other half of the test cases.

For every test case another fault has to be injected into the fault emulating device. Consequently another modified programming file needs to be generated and the fault emulating device needs to be reconfigured for every test case. Thus a new modified synthesizable VHDL-description has to be created. For every test case we insert a saboteur between components or create a mutant of a component of the VHDL-description. This allows injecting a single fault into the fault emulating device, while the golden device remains fault-free. Considered types of faults are stuck-at, bit-flip, stuck-then, stuck-else, assignment control and stuck-data. If the entire system (golden device in cooperation with the fault emulating device) detects the fault and consequently enters and/or maintains the safe state, the test is successful. Otherwise the test fails.

The synthesizable VHDL-description can be parameterized before compilation. The parameters are constants which are defined in a separate VHDL-package. There is one constant per fault insertion test case. If all constants are set to 0, it is not possible to inject a fault into the CPLD after compilation and configuration via the fault enable switch. The synthesized circuit contains no fault injection logic. Thus in the VHDL-description for the golden device, all constants are set to 0. After compilation and configuration not a single logic element is required for mutants or saboteurs in this case.

If the constant for a certain fault injection test case is set to 1, it is possible to inject a corresponding fault after compilation and configuration using the fault enable switch. In this case the synthesized description of the fault emulating device contains a saboteur or a mutant, which can be activated via the fault enable switch. Thus for every fault insertion test case a single constant is set to 1 in the VHDL-description of the fault emulating device. The remaining constants are set to 0. Consequently there is only a little area overhead caused by a little fault injection logic required for a single saboteur or a single mutant for each test case.

There is no guarantee that the mapping of the modified VHDL-description on the logic elements of the CPLD is similar to the mapping of the unmodified VHDL-description on the logic elements. That is why the presented approach is limited to systems which contain at least two channels where each channel contains a CPLD or FPGA, which can be configured independently. There must always be at least one channel that contains a CPLD or FPGA that is configured with the compiled, unmodified VHDL-description.

For every test case a certain fault is injected, when the system is in one of two states. In the first case a fault is injected, when the F-module is in state start-up and in the second case a fault is injected, when the F-module is in state motor running. A test case is successful if the F-module either does not leave state start-up (safe) or if it enters and maintains the safe state.

Following steps are necessary to verify all safety integrity measures:

1. Perform systematic FMEAs on system- and design-level to identify necessary test cases
2. Define expected results for each test case
3. Repeat for every test case
    a. Insert a saboteur into the VHDL-description or create a mutant
    b. Add a VHDL-constant to be able to enable or disable the test case before compilation
4. Repeat for every test case
    a. Edit VHDL-constants to enable injection via the appropriate saboteur or mutant
    b. Compile modified VHDL-description
    c. Reconfigure the fault emulating device
    d. Inject fault when system is in state start-up
    e. Restart to inject fault in state motor running
    f. Decide if test was successful
    g. Analyze and document result

The effort for all fault insertion test cases is dominated by steps 1 and 2, which consume much time for analysis and documentation. The time required for inserting saboteurs or mutants, editing VHDL-parameters, compiling the VHDL-description and reconfiguring the fault emulating device is comparably small.

VI. SYNTHESIS

An example demonstrates how the synthesis results can be influenced by modifying VHDL-constants, to make fault insertion possible. Figure 4 and Figure 5 illustrate schematics of simple combinational circuits, which were synthesized. The circuits have an output that depends on the inputs A_INPUT, B_INPUT, C_INPUT and D_INPUT. If all constants in the corresponding VHDL-package are set to 0, the FAULT_ENABLE pin is not connected to the saboteurs (simple fault injection multiplexers for stuck-at faults). Thus the saboteurs always switch the correct input value to the output (Figure 4). Consequently these saboteurs require no CPLD-resources due to optimizations of the tool, we use for synthesis and fitting.

If the first constant in the corresponding VHDL-package is 1, while the second constant is 0, it is possible to inject a stuck-at-1 fault between the output of the AND-gate A_AND_B and one input of the OR-gate. In this case the FAULT_ENABLE pin is connected to a saboteur. Thus it is possible to inject a stuck-at-1 fault using the fault enable switch (Figure 5). Consequently the synthesized circuit requires an additional logic element.

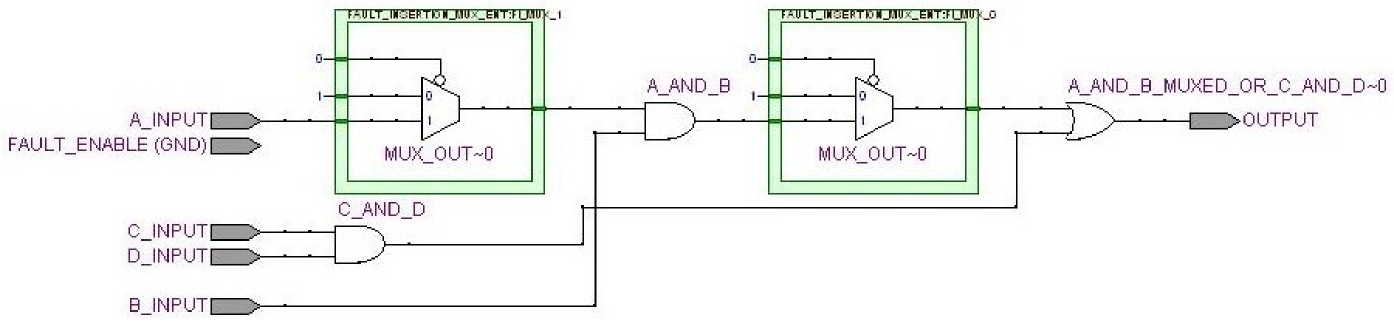

FAULT_INSERTION_MUX_ENT:FI_MUX_1
0
1
A_INPUT
FAULT_ENABLE (GND)
MUX_OUT~0
A_AND_B
FAULT_INSERTION_MUX_ENT:FI_MUX_0
MUX_OUT~0
A_AND_B_MUXED_OR_C_AND_D~0
OUTPUT
C_AND_D
C_INPUT
D_INPUT
B_INPUT

Figure 4. No fault can be injected

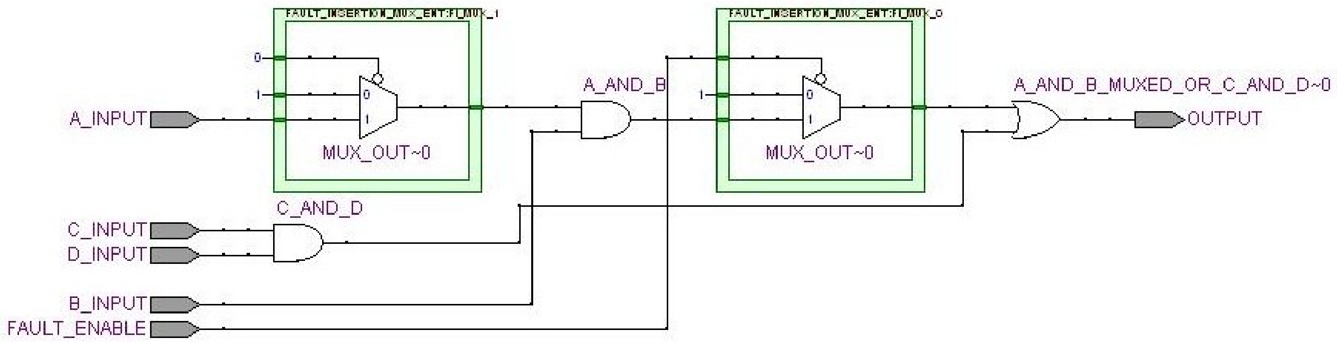

FAULT_INSERTION_MUX_ENT:FI_MUX_1
0
1
A_INPUT
MUX_OUT~0
A_AND_B
FAULT_INSERTION_MUX_ENT:FI_MUX_0
1
MUX_OUT~0
A_AND_B_MUXED_OR_C_AND_D~0
OUTPUT
C_AND_D
C_INPUT
D_INPUT
B_INPUT
FAULT_ENABLE

Figure 5. A stuck-at-1 fault can be injected between the AND-gate and the OR-gate

VII. Experimental Results

The compiled description requires approximately 95% of the 240 logic elements of the used type of CPLD [9]. There remain only a few logic elements, which can be used for fault injection logic. Thus we performed a number of test cases using mutants and saboteurs to prove that the proposed method is applicable. On average the synthesized circuit required 1.72 additional logic elements (less than 1% of the device resources) due to additional fault insertion logic. Although our design already requires approximately 95% of the available logic elements, the method is still applicable.

VIII. Conclusion

The presented method allows the application of a proven verification method (fault insertion testing) for a novel CPLD-based fail-safe system. Methods to verify comparable microcontroller-based systems using fault injection are not applicable for the fail-safe system, because it contains exclusively CPLDs. It is possible to verify the safety integrity measures of the fail-safe system using the system hardware (F-module). The successful verification of the realized safety integrity measures using fault insertion testing is an important prerequisite for the certification of the developed CPLD-based fail-safe system in adherence with the standard IEC 61508.

TÜV SÜD assessed a plan for validation and verification of the fail-safe system. This plan also describes the presented method to perform fault insertion testing. TÜV SÜD stated that the presented method is appropriate.

References

[1] P. Sundaram, and J.G. D'Ambrosio, "Controller integrity in automotive failsafe system architectures," SAE Transactions, 2006, vol. 115, pp. 370–377.

[2] M. Abramovici, A.D. Friedman, and M.A. Breuer, Digital Systems Testing and Testable Design. IEEE PRESS. 1994.

[3] W.L. Gallagher, H.H. Yao, and E.E. Swartzlander Jr., "Fault simulation with PLDs," In Proc. of the 31st Asilomar Conference on Signals, Systems & Computers, 1997, pp. 411–415.

[4] Shih-Arn Hwang, Jin-Hua Hong, and Cheng-Wen Wu, "Sequential circuit fault simulation using logic emulation," IEEE Transations on Computer-Aided Design of Integrated Circuits and Systems, 1998, vol 17, pp. 724–736.

[5] P. Ellervee, J. Raik, K. Tammemäe, and R. Ubar, "Environment for FPGA-based fault emulation,“ In Proc. of the Estonian Academy of Sciences. Engineering, 2006, vol 12, pp. 323–335.

[6] IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems.

[7] J. Gracia, J.C. Baraza, D. Gil, and P.J. Gil, "Comparison and. application of different VHDL-based fault injection techniques," In Proc. of the International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'01), 2001, pp. 233-241.

[8] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson, "Fault injection into VHDL models: the MEFISTO tool.,“ In Proc. of the 24th International Symposium on Fault-Tolerant Computing (FTCS-24), 1994, pp. 66-75.

[9] P. Leventis, B. Vest, M. Hutton, and D. Lewis, "MAX II: A low-cost, high-performance LUT-based CPLD," In Proc. of the Custom Integrated Circuits Conference, 2004, pp. 443–446.

[10] G. Grießnig, C. Steger, and R. Weiß, "CPLD basierende homogen redundante fehlersichere Architektur,“ In Proc. of the Informationstagung Mikroelektronik (ME08), 2008, pp. 201–205.

A CPLD-based Safety Concept for Industrial Applications

Gerhard Grießnig[1,2], Roland Mader[1,2], Christian Steger[2], Reinhold Weiß[2]
[1]AVL List GmbH, Austria
[2]ITI, Graz University of Technology, Austria

***Abstract*—Industry demands cost-efficient approaches for the realization of uncomplex safety functions in industrial automation. Therefore new approaches need to be considered. For this purpose the implementation of safety functions in hardware using CPLDs is an option. This approach does, in contrast to microcontroller-based systems, not require the development of startup- and online tests for RAM and CPU. Therefore efforts for design, implementation and verification of these safety integrity measures can be saved as well as hardware resources for the execution of tests. Based on this idea, a CPLD-based safety concept has been elaborated that allows to realize safety functions by exclusively using CPLDs. The safety concept has been derived from normative safety requirements, functional safety requirements as well as other non-functional requirements. The safety concept comprises a CPLD-based redundant fail-safe system architecture, safety integrity measures and a precise definition of the safe state and the unsafe state of possible target applications. An industrial power drive system is presented that has been enhanced with uncomplex safety functions to increase its safety integrity. These safety functions are able to avoid the application of power to an electric DC motor, if demanded. They were realized by a fail-safe system. This system adopts the CPLD-based safety concept.**

***Index Terms*—safety concept; industrial application; power drive system; fail-safe system; safety function; IEC 61508; CPLD**

I. Introduction

Fail-safe systems achieve and maintain a safe state in case of a failure. Due to this valuable feature this class of systems is often used for safety-critical applications. There this feature is exploited to protect people, the environment or property from harm. To be able to fail safely fail-safe systems need to be able to defect faults during runtime.

Strategies to detect faults during runtime heavily depend on the components a fail-safe system consists of. Components that are frequently used for fail-safe systems are microcontrollers. These devices require the implementation of safety integrity measures like online- and startup tests for CPU and RAM (see e.g. [1]) to ensure their proper operation. Among others, in the field of power drive systems for industrial automation, fail-safe systems are asked to be able to carry out safety functions that are comparably simple. In this case the realization of complex safety integrity measures for RAM and CPU is a burden, because a lot of the efforts are spent on the design, implementation and verification of the safety integrity measures and a lot of hardware resources (e.g. RAM, Flash, computing time) are used to perform these tests. Clearly, this affects the development costs and the hardware costs of these systems.

Therefore alternative, cost-efficient concepts are demanded by the industry. An alternative concept is the use of a pure hardware implementation of safety functions and safety integrity measures based on CPLDs. With this approach, the use of RAM and CPU tests is superfluous as neither CPU nor RAM are present. Obviously for this approach a proper safety concept is required that allows to safely realize CPLD-based safety functions in adherence to the rigorous demands of safety standards that apply for industrial automation.

This work contributes by presenting a safety concept for a class of applications in the field of power drive systems for industrial automation. This safety concept adequately addresses normative and functional requirements of relevant target applications as well as other non-functional requirements. It comprises a definition of a suited, redundant system architecture for the implementation of a class of simple safety functions on CPLDs, defines safety integrity measures to detect faults during startup and runtime and precisely defines the safe state and the unsafe state of target applications.

An industrial application is presented. This application is a power drive system for industrial automation. The power drive system controls an electric DC motor and converts three phase current to direct current. The CPLD-based safety concept has been adopted to realize a fail-safe system that enhances the power drive system with two safety functions. This fail-safe system is an optional module that can be plugged into the power drive system to increase its safety integrity.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the CPLD-based safety concept for power drive systems in industrial automation. Section IV describes a safety-critical industrial application that adopts the CPLD-based safety concept. Finally Section V concludes.

II. Related Work

In [2] architectures for microcontroller-based fail-safe systems and concepts to achieve safety integrity are described. Described concepts are single-controller strategy for low levels of safety integrity, asymmetric controller strategy for moderate levels of safety integrity and symmetric controller for higher levels of safety integrity.

The single-controller strategy comprises a single microcontroller, which performs periodic self tests to detect faults. To increase safety integrity the microcontroller needs to reset an additional watchdog periodically. If a fault is detected or the watchdog is not reset, a safe state is achieved and maintained.

The second concept (asymmetric controller strategy) comprises a primary microcontroller and an intelligent watchdog (secondary microcontroller or ASIC). The intelligent watchdog requests diagnostic checks from the primary microcontroller periodically. Communication timeouts or detected faults cause the transition into a safe state.

In contrast the symmetric controller strategy uses two identical microcontrollers. These controllers carry out the same program and compare the results of their computations periodically to detect faults (e.g. via CAN bus). Communication timeouts or detected faults lead to a safe state.

In [1] software-failsafe techniques are described that allow achieving safety integrity in microcontroller-based fail-safe systems. The authors describe how preliminary hazard analysis (PHA), failure modes and effects analysis (FMEA), fault tree analysis (FTA) and fault coverage matrix can be applied to determine safety integrity measures that need to be applied in order to ensure safety. Surveyed safety integrity measures are suited to check for RAM failures, CPU failures, software processing errors, interface failures or communication failures.

In [3] a TMR (triple mode redundancy) system is described, which exclusively consists of four PLDs, where one PLD contains diagnosis circuitry. According to the authors, most of the faults occur in the input and output interfaces of safety-critical systems. Therefore they utilize the concepts 'safe input cell' and 'safe output cell', which are intended to be integrated into the PLDs.

A safe input cell receives inputs from three sensors, which measure the same physical quantity. A voter circuit determines the probably correct value of the physical quantity. Safe output cells represent the logic value 1 by a periodic signal, while the logic value 0 is represented by a constant signal. Safe input cells and safe output cells comprise an additional state machine, circuitry for the detection of failures and circuitry for the signaling of errors to other system components.

The authors of [4] discuss common mode failures (CMF) in redundant systems with focus on VLSI systems. They refer to the term common mode failure as the result of an event, which, because of dependencies, causes a coincidence of failure states of components in two or more separate channels of a redundancy system, leading to the defined system failing to perform its intended function. The authors use the terms 'common mode failure' and 'common cause failure' interchangeably.

According to the authors, CCF can occur in hardware or software. They can be caused by permanent faults (e.g. bugs) or intermittent faults (e.g. weak signals) introduced during the development process of a redundant system. Exemplary faults are ambiguous specifications, bugs of used tools, incomplete verification, manufacturing defects or non-exhaustive testing. CCF can also be caused by external disturbances during the operation of the redundant system. Caused effects can reside transiently or permanently.

Techniques to handle common-cause failures comprise CMF avoidance (use of mature and verified components, conformance to standards, use of formal methods, use of design automation, implementation of design rules, diverse realization of redundant components), CMF removal (design reviews, simulation, verification, testing and fault-injection) and CMF tolerance (watchdog timers, exception handlers, runtime checks, concurrent error detection).

In [5] fault-tolerant architectures including multiple CPUs for the integration on a single chip are described. The authors review the architectures lock-step, loosely synchronized dual-processor and triple modular redundant (TMR). Moreover, they propose new multi-CPU architectures for the integration on a single chip. The authors point out the high susceptibility of single-chip implementations to common-cause failures if no special measures are undertaken. Nevertheless, they consider on-chip redundancy to be an important design practice due to the permanent faults caused by increasingly shrinking integration densities of chips.

In [6] an approach to reduce the probability of common cause failures called faultRobust is described. This approach is based on a library for SoC (system on chip) design that contains components for hardware-centric fault detection and fault tolerance. The components allow to achieve safety integrity by performing memory protection, CPU diagnostic checking as well as system bus- and interface supervision. A separate bus is used for communication between diagnostic units. Functionally and architecturally diverse components contribute to reducing the probability for common cause failures.

A system FMEA methodology for SoC design according to the faultRobust approach is described in [7]. The methodology can be used to assess the safe failure fraction (SFF) of a SoC.

III. Novel CPLD-based Safety Concept

A. Requirements to the Safety Concept

A safety concept is heavily influenced by normative safety requirements as well as requirements of possible target applications. Also other requirements concerning availability of space and costs need to be considered.

An important property of fail-safe systems is their *ability to detect faults* during runtime to be able to decide when a *safe state* needs to be achieved and maintained. Therefore the capability to detect faults and to achieve and maintain a safe state is an essential requirement that needs to be taken into account during the elaboration of a safety concept. This requirement obviously impacts the system architecture.

Depending on the application area, *safety standards* need to be considered. Target applications for this safety concept are electric drives for highly critical applications in the field of industrial automation. Therefore requirements of the safety standards IEC 61508 [8], IEC 61800-5-2 [9] and EN ISO 13849 [10] apply. Of particular importance is a requirement of EN ISO 13849 that constrains that a single fault must not cause the loss of the safety functions for the intended category 4 and performance level e. Therefore a HFT (hardware fault tolerance) of 1 shall be achieved and a single fault has to be detected at the time when a safety function is demanded or

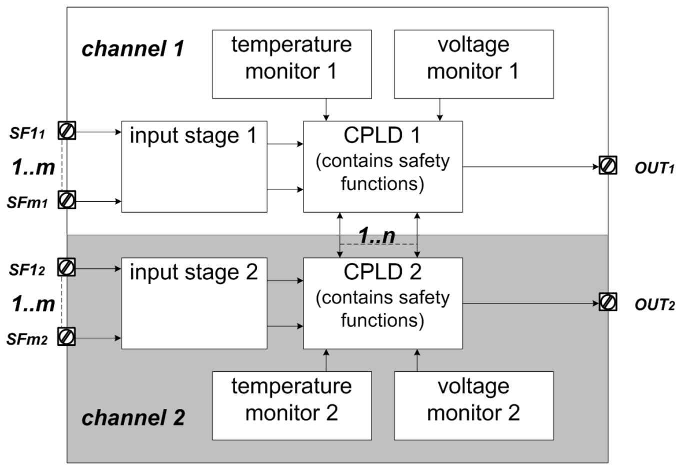


Fig. 1. System Architecture

earlier. This requirement clearly impacts the system architecture.

The kind of *safety functions* that shall be realizable need to be considered as well as its interacting actuators. The safety functions that shall be realizable for possible target applications need the ability to avoid that power is applied to the motor of an electric drive in different manners. Therefore safety functions are not intended to control torque or speed of a motor during its operations but they shall be able to avoid the motor's operation, if demanded. This requirement influences the interfaces of the system architecture and safety integrity measures.

Costs are also relevant for a system architecture that shall be applied in industrial practice. Obviously, since the system architecture is designed in order to increase safety, costs are a secondary factor. To tackle the challenge of being safe and as cost efficient as possible, for the design of the system architecture preferably COTS (components of the shelf) shall be used. Moreover the safety functions and safety integrity measures shall be efficiently realizable to demand as less CPLD device resources as possible.

An important factor is the size of the resulting PCB (printed circuit board). Depending on the target applications, *physically available space* is limited. Therefore besides the use of technologies like multi layer SMD boards, FBGA CPLDs as well as computer-aided layout tools for hardware development, it is required to design the system architecture with respect to required PCB space.

B. System Architecture

A key issue for a safety concept is the system architecture of safety-critical embedded systems that are designed and implemented accordingly to the safety concept. The derived requirement for HFT=1 clearly states the necessity for two separate channels where each channel is able to redundantly carry out the safety functions. Moreover the requirement to control and detect a single fault on demand or earlier makes it necessary to place monitors on each channel. Therefore the system architecture must contain one temperature monitor per channel that can detect if the system temperature is within particular limits as well as one voltage monitor per channel that can detect if the supply voltage is within particular limits. This allows the achievement of a safe state in case of dangerous operational conditions.

To allow to demand safety functions for each channel separately, the system architecture shall contain one terminal per channel for each safety function ($SF1_1$-$SF1_m$ and $SF2_1$-$SF2_m$). The terminals shall be connected to the CPLD-circuitry via optocouplers to ensure electrical isolation. To be able to achieve and maintain a safe state, each channel shall contain a terminal that allows to prevent or allow that power is applied to the motor of the electric drive (OUT_1-OUT_2).

As stated above, besides safety-related requirements also requirements concerning PCB and costs are relevant to the system architecture. Therefore it is inviting to use a single CPLD that is shared by both channels (on-chip redundancy). This CPLD could contain redundant implementations of safety functions and safety integrity measures.

The drawback of a single-CPLD approach is its susceptibility to common cause failures. According to [8] the term common cause failure refers to a failure of the entire system caused by one or more events, which cause the failures of two or more redundant channels.

In case of a single-CPLD system architecture the redundant implementations of safety functions and safety integrity measures on a single CPLD are likely to fail because of the same occasion. Examples for such occasions are a defective silicone substrate, a defective power distribution network, a damaged package or a corrupted CPLD-configuration of the single CPLD. This would cause the loss of both redundant implementations of the safety functions.

Therefore an approach that is more insusceptible to common cause failures needs to be considered. A promising approach is a system architecture that comprises two CPLDs, where each CPLD contains an independent implementation of safety functions and safety integrity measures. In this case independent and physically separated silicone substrates, power distribution networks, packages and CPLD-configurations are present for the two channels of the system. To ensure that both CPLDs are able to detect faults, they are connected by wires that allow the exchange of signals. Therefore disadvantages in terms of PCB size and costs need to be accepted to ensure the applicability of the system architecture for safety-critical applications. The resulting system architecture is depicted by Figure 1.

C. Safety Integrity Measures

The applied safety concept defines a number of safety integrity measures that are executed by both CPLDs concurrently to detect faults during startup and runtime. Once a fault is detected the safe state needs to be achieved and maintained regardless of the system's operation mode.

- Startup Test: A startup test needs to be carried out each time the system is switched on to ensure that both CPLDs are able to carry out the entire safety functions properly.

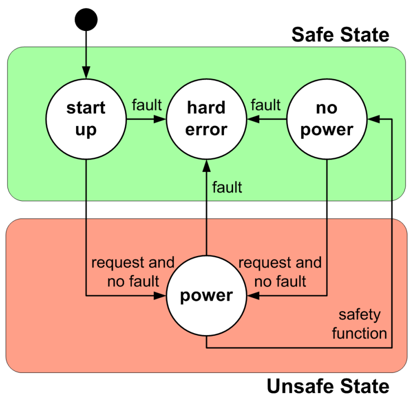


Fig. 2. Safe State, Unsafe State and Possible Transitions

- Background Test: A test is required that continuously tests the components that are necessary to carry out the safety functions. This test needs to be carried out as long as the system is working.
- Discrepancy Test: The both CPLDs are connected by wires that allow the exchange of signals. These signals need to be representative for the state of the safety functions. If corresponding signals of the CPLDs are discrepant for more than a certain time, a fault is assumed.
- Demand Test: When a safety function is carried out this test needs to find out if the implementations of the safety functions still meet the real time requirements of the application. Moreover this test needs to be able to detect short circuits between the terminals that can be used to ensure that no power can be applied by the motor.
- Monitoring: The voltage monitor and the temperature monitor are connected to the CPLD of each channel to be able to signal faults.

D. System States

An important point when a safety concept is defined is the definition of the safe state and the unsafe state of the system. The definition of the safe and the unsafe state is clearly derived from the considered class of power drive system applications. In this case the *safe state* is achieved if no power can be applied to the electric motor. The safety concept requires the safety-critical terminals of the system architecture that are connected to an electric motor to be switched to GND. In the *unsafe state*, power can be applied to the motor and the motor can be controlled by the application. In this case the safety-critical terminals need to switched to V_{DD}. While the safe state can be divided in substates, the unsafe state is atomic.

After power on, the state *start up* (safe) needs to be entered. In this state initializations and parameterizations need to be performed. Moreover Startup Test, Discrepancy Test and Monitoring need to be performed. If explicitly required by the application and no fault is detected, the unsafe state can be entered to allow the application to control the electric motor. During this state Background Test, Discrepancy Test, Monitoring and Demand Test need to be performed.

If demanded, the safety functions are carried out in the unsafe state. If the execution of the functions is finished, the safe state *no power* (safe) is entered. In this state Monitoring and Discrepancy Test are carried out. If explicitly required by the application and no fault is detected, the unsafe state can be entered again.

Whenever a fault is detected, the state *hard error* (safe) is entered. This state must not be left any more as long as the system is running. The states, substates and transitions are depicted by Figure 2.

IV. Industrial Application

The applicability of the elaborated safety concept has been evaluated using an industrial power drive system. This power drive system is able to control a DC motor. It converts three-phase current to direct current for low-power and high-power DC motors. The power drive system contains a microcontroller and a DSP (digital signal processor). These two devices realize standard functions that are able to control the DC motor in different manners. If the power drive system is not able to carry out these functions any more due to a failure, a controlled motor cannot be stopped any more and is able to harm people (e.g. worker in a factory). This is a risk. To reduce this risk and to make the power drive system applicable for even highly safety-critical applications, safety functions need to be realized accordingly to SIL 3 in adherence to IEC 61508. These safety functions reduce the risk of failing standard functions significantly. The safety functions are defined by IEC 61800-5-2 and have been implemented using a CPLD-based fail-safe system based on the safety-concept described by Chapter III. The following two safety functions are implemented:

- Safe Torque Off (STO): Power, that can cause rotation (or motion in the case of a linear motor), is not applied to the motor. The PDS(SR) will not provide energy to the motor which can generate torque (or force in the case of a linear motor).
- Safe Stop 1 (SS1): The PDS(SR) initiates the motor deceleration and initiates the STO function after an application specific time delay.

A. Use Case: Power Drive System (DC-Converter)

The application block diagram in Figure 3 depicts the main components of the PDS including the fail-safe system. The architecture of the power drive system is modular. There is a clear separation between the control electronic which is implemented on the Power Interface and the power electronic which is located on the Power Stage.

Moreover the CPLD-based fail-safe system that is able to carry out the safety functions is an optional module. If no safety functions are required due to a low criticality of the application the power drive system is used for, it is able to work without the CPLD-based fail-safe system. In this case the safety-relevant output signals STO_1 and STO_2 are permanently connected to V_{DD} and power can always be applied by the motor. If the power drive system is used for a highly critical application, the fail-safe system can be plugged into the power

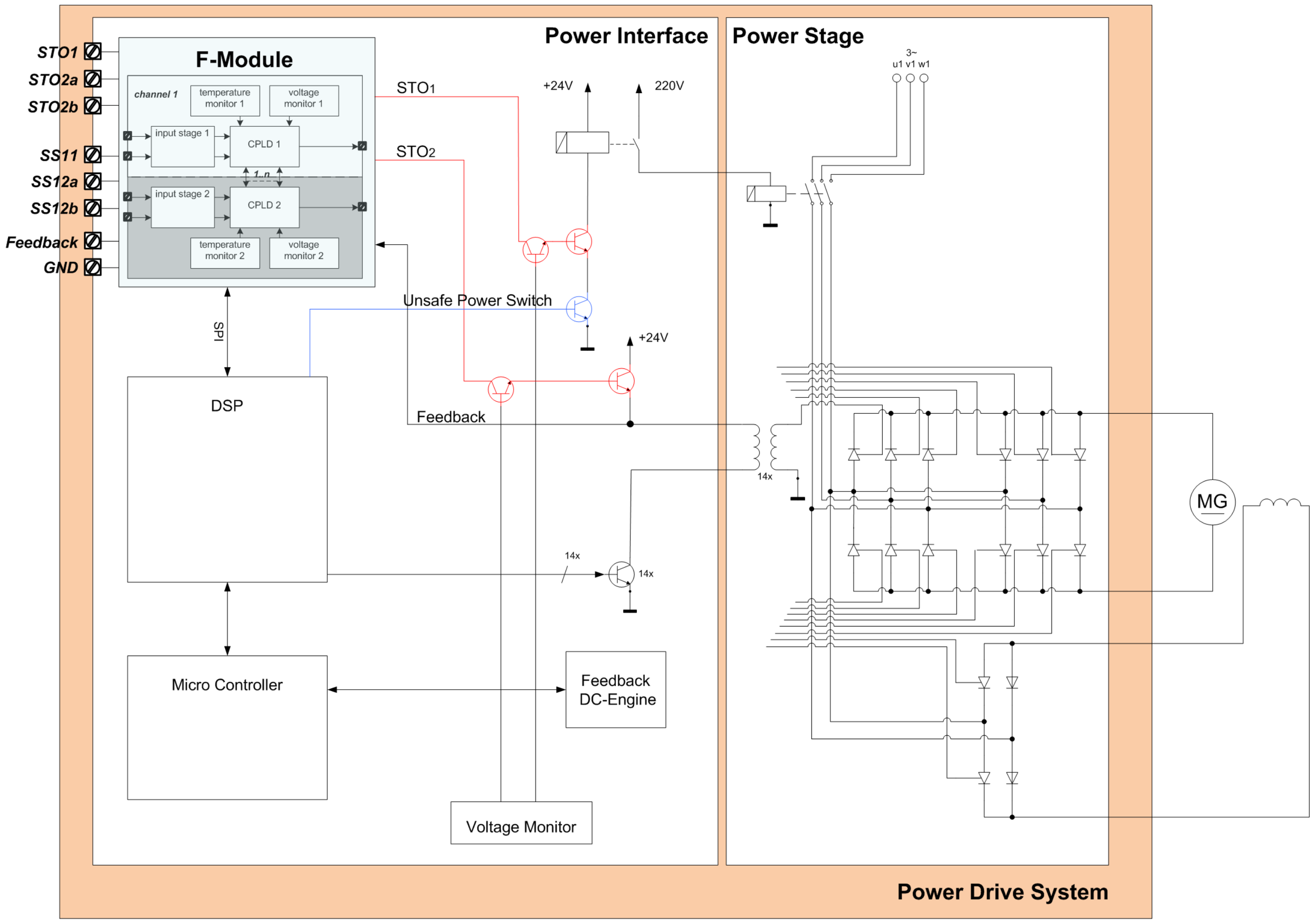


Fig. 3. Power Drive System

converter to ensure a high level of safety integrity. In this case the safety-relevant output signals STO_1 and STO_2 are switched by the fail-safe system. The power drive system consists of the following components:

- Microcontroller: The microcontroller is the dominant component of the PDS system. It provides an interface to the user and to the higher level infrastructure. Moreover it continuously calculates the required voltage at the terminals of the DC motor depending on measured quantities of the DC engine and transmits the value of the required voltage to a DSP.
- DSP: The DSP is necessary to switch the thyristor bridges in real time to convert the three-phase voltage into the requested direct voltage.
- Voltage Monitor: A voltage monitor is necessary to protect the PDS system from destruction by overvoltage.
- Thyristor Bridge: The thyristor bridge is located on the power stage for voltage conversion.
- Transistor Switches: The transistor switches are controlled by the DSP to switch the thyristors. These switches are located at the power interface. They are electrically decoupled from the power stage.
- Feedback DC Engine: To reduce the calculation effort of the microcontroller, an ASIC is used to perform precalculations concerning measured electric fields, currents and voltages.
- F-Module: The F-Module is the fail-safe system that adopts the CPLD-based safety concept and realizes the safety functions accordingly to the applied standards.

B. *F-Module*

If plugged into the PDS, the F-Module communicates with the microcontroller via the DSP. Therefore a SPI (serial peripheral interface) is established between DSP and F-module. The DSP alternatingly exchanges messages with the two CPLDs of the F-module. Whenever a CPLD receives a message from the DSP, it resets an internal counter. If this counter expires, a DSP failure is assumed and a safe state achieved. Therefore the CPLDs act as watchdogs for the DSP. Moreover the SPI messages are used to inform the DSP and the microcontroller about the state of the F-module and to inform the F-module if a change of its state is requested (e.g. *from startup* to *power*).

The SPI is also used for the parametrization of the safety function SS1 in state *startup*. In this state, the microcontroller

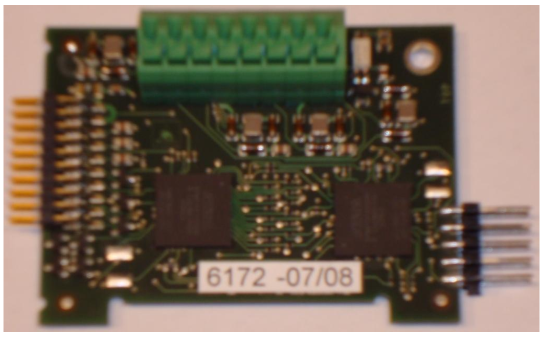


Fig. 4. F-Module

transmits parameters via the DSP to each CPLD. The parameterization allows to vary the application specific time delay of the safety function SS1 in *100ms* steps between *100ms* and *5min*. The possibility to parametrize SS1 is an important feature and it extends the range of possible applications significantly. Many applications originating from the high power segment deal with very high currents. In these cases, a prompt turn-off of a machine without reducing the current before can damage expensive switches. Furthermore an immediate turn-off can harm the production equipment or can destroy the products of the production process. Therefore an application specific time delay is required to inform the microcontroller via the DSP about the demand of a safety function to give the PDS time to reduce currents and enable the high level infrastructure to ramp down the production process before a safe state is achieved.

The F-Module monitors the status of the of the two STO channels of the power drive system via feedback signals. The first feedback signal provides the status of the DC motor power switch and the second feedback signal provides the enable status of the thyristor bridge. If no enable status is available, no thyristor can be ignited and consequently no power can be applied to the DC motor.

A hardware prototype of the F-Module is depicted in Figure 4. For the prototype of the F-module Altera MAX II CPLDs were used [11]. The F-Module has dedicated terminals (on top of the figure) to demand the safety functions with hardwired switches or via digital safety outputs of the higher level infrastructure. The terminals on the left side of the figure are dedicated to power supply of the F-module and to communication with the DSP via SPI.

Requirements of a draft of IEC 61508 to the development process of safety-critical CPLDs have been considered. Therefore the prototype has been developed accordingly to a V-model using an appropriate tool chain [12]. For the verification of the realized safety integrity measures a new fault insertion testing technique [13] has been applied.

V. Conclusion

This work was motivated by the fact that industry demands cost-efficient concepts for the realization of safety functions for power drive systems in industrial automation. Therefore a CPLD-based safety concept has been elaborated that does not require the realization of microcontroller-specific software safety integrity measures like RAM tests or CPU tests that require a lot of effort in terms of design, implementation and verification as well as hardware resources (e.g. RAM, Flash, computing time). The applicability of this safety concept has been proofed in terms of safety and cost-efficiency by its utilization for an industrial power drive application. The safety concept and its application were accessed by an independent certification authority (TÜV SÜD). TÜV SÜD stated that the presented concept is suited to achieve SIL 3 in adherence to IEC 61508 and IEC 61800-2 as well as Cat 4, PL e in adherence to EN ISO 13849.

It can be concluded the the use of a CPLD-based safety concept is a competitive alternative to the use of microcontroller-based safety concepts, if comparably uncomplex safety functions need to be realized. In this case the comparably simple functionality does not justify a software implementation including the great effort for development of complex software safety integrity measures to make the use of microcontrollers acceptably safe.

References

[1] E. G. Leaphart, B. J. Czerny, J. G. D'Ambrosio, C. L. Denlinger, and D. Littlejohn, "Survey of Software Failsafe Techniques for Safety-Critical Automotive Applications," in *Proc. of the 2005 SAE World Congress*. SAE, Apr. 2005, pp. 1–16.

[2] P. Sundaram and J. G. D'Ambrosio, "Controller Integrity in Automotive Failsafe System Architectures," *SAE Transactions*, vol. 115, pp. 370–377, 2006.

[3] J. Alvarez, J. Marcos, and S. Fernandez, "Safe PLD-based Programmable Controllers," in *Proc. of the International Conference on Field Programmable Logic and Applications*, vol. 2, Aug. 2005, pp. 559–562.

[4] S. Mitra, N. R. Saxena, and E. J. McCluskey, "Common-Mode Failures in Redundant VLSI Systems: A Survey," *IEEE Transactions on Reliability*, vol. 3, pp. 285–295, 2000.

[5] M. Baleani, A. Ferrari, L. Mangeruca, A. L. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-tolerant Platforms for Automotive Safety-Critical Applications," in *Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES'03)*. ACM, Nov. 2003, pp. 170–177.

[6] R. Mariani and P. Fuhrmann, "Comparing fail-safe microcontroller architectures in light of IEC 61508," in *Proc. of the 22nd International Symposium on Defect and Fault Tolerance in VLSI Systems*. IEEE, Sep. 2007, pp. 123–131.

[7] R. Mariani, G. Boschi, and F. Colucci, "Using an innovative SoC-level FMEA methodology to design in compliance with IEC61508," in *Proc. of the Design, Automation & Test in Europe Conference and Exhibition (DATE'07)*. IEEE, Apr. 2007, pp. 1–6.

[8] IEC, "IEC 61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, part 1-7," 2002.

[9] ——, "IEC 61800-5-2, Adjustable Speed Electrical Power Drive Systems," 2005.

[10] ISO, "EN ISO 13849, Safety of Machinery, part 1-2," 2006.

[11] P. Leventis, B. Vest, M. Hutton, and D. Lewis, "MAX II: A Low-Cost, High-Performance LUT-Based CPLD," in *Proc. of the Custom Integrated Circuits Conference*. IEEE, Oct. 2004, pp. 443–446.

[12] G. Grießnig, R. Mader, C. Steger, and W. Reinhold, "Design and Implementation of Safety Functions on a Novel CPLD-based Fail-Safe System Architecture," in *Proc. of the 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS'10)*. IEEE, Mar. 2010, pp. 202–212.

[13] ——, "Fault Insertion Testing of a Novel CPLD-based Fail-Safe System," in *Proc. of the conference on Design, Automation & Test (DATE'09)*. IEEE, Apr. 2009, pp. 214–219.

Bibliography

[ABD$^+$07] EGAS Working-Group (Audi, BMW, Daimler, Porsche, and VW). Standardized E-Gas monitoring concept for engine management systems of gasoline and diesel engines, 2007. Also available under http://wenku.baidu.com/view/aedb0922bcd126fff7050b51.html.

[AG07] SIEMENS AG. Safety Integrated - Terms and Standards Terminologie in der Maschinensicherheit, 2007. Also available as https://www.automation.siemens.com/mcms/safety-integrated/de/Seiten/funktionale-sicherheit.aspx.

[AG10] SIEMENS AG. SIMOREG DC-MASTER, 2010. Also available as http://www.automation.siemens.com/mcms/infocenter/dokumentencenter/ld/Documentsu20Brochures/dc-stromrichter/ws-dc-master-en.pdf.

[AGZ$^+$11] E. Armengaud, G. Grießnig, M. Zoier, D. Chen, M. Biehl, C. Hein, T. Ritter, A. Baumgart, and R. Tavakoli Kolagari. Model-based Toolchain for the Efficient Development of safety-relevant Automotive Embedded Systems. In *SAE World Congress*, 2011.

[ALRL04] A. Avizienis, JC. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, March 2004.

[Alt08] Altera. Altera Onlineshop, September 2008. Also available as http://www.buyaltera.com/scripts/partsearch.dll?PV-6=8.

[Alt10] Altera. Quartus II Version 10.1 Handbook, 2010. Also available as http://www.altera.com/.

[AMF05] J. Alvarez, J. Marcos, and S. Fernandez. Safe PLD-based Programmable Controllers. In *Proc. of the International Conference on Field Programmable Logic and Applications*, volume 2, pages 559–562, August 2005.

[ATE10] ATESST2 Project Consortium. EAST-ADL Domain Model Specification, 2010. Version 2.1, Release Candidate 3.

[BFM$^+$03] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini. Fault-tolerant Platforms for Automotive Safety-Critical Applications. In *Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES'03)*, pages 170–177.

ACM, November 2003. Also available as http://embedded.eecs.berkeley.edu/Respep/Research/asves/paper2003/Baleani_cases03.pdf.

[BGA+10] J.P. Blanquart, G. Grießnig, E. Armengaud, M. Cifaldi, M. Fortes da Cruz, T. Gross, G. Jolliffe, F. Pouzolz, N. Priggouris, M. Shawky, and M. Törngren. D_SP1_R5.8_M2: Survey of state-of-the-practice and state-of-the-art in safety and diagnosability V2, December 2010. Also available under: https://cesarproject.eu/.

[BHU08] J. Börcsök, A. Hayek, and M. Umar. Implementation of a 1oo2-RISC-Architecture on FPGA for Safety Systems. In *Proc. of the IEEE/ACS International Conference on Computer Systems and Applications*, pages 1046–1051. IEEE, April 2008.

[Bla08] B.S. Blanchard. *System Engineering Management.* John Wiley and Sons Inc., 2008.

[BMSS00] C. Bolchini, R. Montandon, F. Salice, and D. Sciuto. Design of VHDL-Based Totally Self-Checking Finite-State Machine and Data-Path Descriptions. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8:98–103, February 2000.

[Bör07] J. Börcsök. *Functional Safety: Basic Principles of Safety-Related Systems.* Hüthig, 2007.

[CB10] P. Conmy and I. Bate. Component-Based Safety Analysis of FPGAs. *IEEE Transactions on Industrial Informatics*, 6:195–205, May 2010.

[CFJ+08] P. Cuenot, P. Frey, R. Johansson, H. Lönn, M.-O. Reiser, D. Servat, R. Tavakoli Kolagari, and D.J. Chen. Developing Automotive Products Using the EAST-ADL2, an AUTOSAR Compliant Architecture Description Language. *Ingénieurs de l'Automobile*, 21:498–516, 2008.

[Cle09] J.R. Clegg. Arguing the safety of FPGAs within safety critical systems. In *Proc. 4th IET International Conference on System Safety 2009. Incorporating the SaRS Annual Conference*, 2009.

[Com02] International Electrotechnical Commission. IEC 61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, part 1-7, 2002. Also available under http://www.iec.ch/.

[Com05a] International Electrotechnical Commission. IEC 60204-1, Safety of machinery - Electrical equipment of machines - Part 1: General requirements , 2005. Also available under http://www.iec.ch/.

[Com05b] International Electrotechnical Commission. IEC 61800-5-2, Adjustable Speed Electrical Power Drive Systems, 2005. Also available under http://www.iec.ch/.

[Com10] International Electrotechnical Commission. IEC 61508:2010 Second Edition, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems 2010(Second Edition), November 2010. Also available under `http://www.iec.ch/`.

[Con08] ATESST Project Consortium. EAST-ADL2 specification, 2008. See `http://www.atesst.org/`.

[Con09] CESAR Project Consortium. CESAR project, 2009. See `http://www.cesarproject.eu`.

[Con11] MBAT Consortium. Technical Annex MBAT - Combined Model-based Analysis and Testing of Embedded Systems, 2011.

[EJ09] C. Ebert and C. Jones. Embedded software: Facts, figures, and future. *IEEE Computer*, 42(4):42–52, 2009.

[ERTU06] P. Ellervee, J. Raik, K. Tammemäe, and R. Ubar. Environment for FPGA-based fault emulation. In *Proc. of the Estonian Academy of Sciences. Engineering*, volume 12, pages 323–335, September 2006. Also available as `http://www.kirj.ee/public/va_te/eng-2006-3_2-6.pdf`.

[ES09] C. Ebert and J. Salecker. Embedded software technologies and trends. *IEEE Software*, 26(03):14–18, 2009.

[EXI10] EXIDA. Position Paper on IEC 61508 2010 - Definition Regarding Minimum HFT / Architectural Constraints, 2010. Also available as `http://www.exida.com/images/uploads/exida_Position_on_IEC_61508_2010_definitions_minimum_HFT_v4.pdf`.

[fS06] International Organization for Standardization. EN ISO 13849, Safety of Machinery, part 1-2, 2006. Also available under `http://www.iso.org/iso/home.html`.

[GBGG01] J. Gracia, J.C. Baraza, D. Gil, and P.J. Gil. Comparison and. application of different VHDL-based fault injection techniques. In *In Proc. of the International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 233–241. DFT01, 2001.

[GFH$^+$09] G. Grießnig, W. Fuchs, F. Hackl, W. Hofmüller, G. Hörist, F. Mossburger, R. Schmid, H. Schwarzmann, and J. Wahrbicher. Method for actuating a DC machine - PatentID WO2009/080384 A1, 2009. SIEMENS AG.

[GKA$^+$] G. Grießnig, I. Kundner, E. Armengaud, S. Torchiaro, and D. Karlsson. Improving automotive embedded systems engineering at european level. *E&I Elektrotechnik und Informationstechnik OVE-Verbandszeitschrift (accepted).*

[GM08] G. Grießnig and R. Mader. V&V Plan, 2008. Hint: Internal Siemens Document.

[GMP$^+$10] G. Grießnig, R. Mader, T. Peikenkamp, B. Josko, M. Törngren, and E. Armengaud. CESAR: Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems. In *Embedded World 2010- ARTEMIS Session*, 2010.

[GMSW09] G. Grießnig, R. Mader, C. Steger, and R. Weiß. Fault Insertion Testing of a Novel CPLD-based Fail-Safe System. In *Proc. of the conference on Design, Automation & Test (DATE'09)*, pages 214–219. IEEE, April 2009.

[GMSW10a] G. Grießnig, R. Mader, C. Steger, and R. Weiß. A CPLD-based Safety Concept for Industrial Applications. In *Proc. of the IEEE - Industrial Symposium on Industrial Electronics (ISIE 2010)*, pages 3027–3032. IEEE, June 2010.

[GMSW10b] G. Grießnig, R. Mader, C. Steger, and R. Weiß. Design and Implementation of Safety Functions on a Novel CPLD-based Fail-Safe System Architecture. In *Proc. of the 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS'10)*, pages 202–212. IEEE, March 2010.

[Gri08] G. Grießnig. Safety Requirements Specification SIMOREG-plus, 2008. Hint: Internal Siemens Document.

[GSW08] G. Grießnig, C. Steger, and R. Weiß. CPLD basierende homogen redundante fehlersichere Architektur. In *Proc. of the Informationstagung Mikroelektronik (ME 2008)*, pages 201–205, October 2008.

[GYJ97] W. Gallagher, H. Yao, and E. Swartzlander Jr. Fault Simulation With PLDs. In *Proc. of the 31st Asilomar Conference on Signals, Systems & Computers*, pages 411–415. IEEE, November 1997.

[Hau06] M. Hause. The SysML Modelling Language. In *Proc. of the 5th European Systems Engineering Conference*, September 2006.

[Ins10] Texas Instruments. Safety Manual TMS570LS20216S Device, January 2010. Also available as http://www.ti.com/.

[Int10] International Organization for Standardization. ISO/FDIS 26262 Road vehicles - Functional safety, 2010. Also available under http://www.iso.org/iso/home.html.

[JAR$^+$94] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson. Fault injection into VHDL models: the MEFISTO tool. In *In Proc. of the 24th International Symposium on Fault-Tolerant Computing*, pages 66–75. FTCS-24, 1994.

[JWR07] P. Jesty, D. Wardt, and R. Rivettl. Hazard Analysis for Programmable Automotive Systems. In *Proc. of the System Safety, 2007 2nd Institution of Engineering and Technology*, 2007.

[KSS09] V. Kharchenko, O. Siora, and V. Sklyar. Design and testing technique of FPGA-based critical systems. In *Proc. 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics*, 2009.

[LF08] H. Lönn and R. Freund. Automotive Architecture Description Language. In *Automotive Embedded Systems Handbook*, chapter 9. CRC Press, 2008.

[LPP10] P. Löw, R. Pabst, and E. Petry. *Funktionale Sicherheit in der Praxis - Anwendung von DIN EN 61508 und ISO/DIS 26262 bei der Entwicklung von Serienprodukten*. dpunkt.verlag GmbH, 2010. isbn: 978-3-89864-570-6.

[LVHL04] Paul Leventis, Brad Vest, Michael Hutton, and David Lewis. MAX II: A Low-Cost, High-Performance LUT-Based CPLD. In *Proc. of the Custom Integrated Circuits Conference*, pages 443–446. IEEE, October 2004.

[Mad08] R. Mader. Diplomarbeit: Entwurf und Implementierung von Sicherheitsfunktionen in konfigurierbarer Hardware, 2008. TU-Graz, Institute for Technical Informatics.

[MBC07] R. Mariani, G. Boschi, and F. Colucci. Using an innovative SoC-level FMEA methodology to design in compliance with IEC61508. In *Proc. of the Design, Automation & Test in Europe Conference and Exhibition (DATE'07)*, pages 1–6. IEEE, April 2007.

[MF07] R. Mariani and P. Fuhrmann. Comparing fail-safe microcontroller architectures in light of IEC 61508. In *Proc. of the 22nd International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 123–131. IEEE, September 2007. Also available as http://ieeexplore.ieee.org/iel5/4358358/4358359/04358380.pdf.

[MGA+11] R. Mader, G. Grießnig, E. Armengaud, A. Leitner, c. Kreiner, Q. Bourrouilh, C. Steger, and R. Weiß. A Computer-Aided Approach to Hazard Analysis for Automotive Embedded System. In *18th IEEE International Conference on the Engineering of Computer Based Systems (ECBS'2011, accepted)*, 2011.

[Mos10] J. Mossinger. Software in Automotive Systems. *IEEE Software Magazine*, 27:92–94, 2010.

[MSM00] S. Mitra, N. R. Saxena, and E. J. McCluskey. Common-Mode Failures in Redundant VLSI Systems: A Survey. *IEEE Transactions on Reliability*, 3:285–295, 2000.

[Nan95] Nancy G. Leveson. *Safeware: system safety and computers*. Addison-Wesley Publishing Company, 1995. isbn: 0-201-11972-2.

[OMG10] OMG. OMG Systems Modeling Language (OMG SysML), V1.2, 2010. Also available as http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf.

[Pol10] Pollux. Process Oriented Electrical Control Units for Electrical Vehicles Developed on a multi-system real-time embedded platform, 2010. See http://www.artemis-pollux.eu/.

[RW10] F. Reichenbach and A. Wold. Multi-core Technology - Next Evolution Step in Safety Critical Systems for Industrial Applications. In *Proc. of the IEEE/13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 339–346. IEEE, 2010.

[SD06] P. Sundaram and J. D'Ambrosio. Controller Integrity in Automotive Failsafe System Architectures. *SAE Transactions*, 115:370–377, 2006.

[SHM05] A. Söderberg, J. Hérard, and L. Bo Mortensen. Guideline for Design and Safety Validation of Safety-Critical Functions Realized with Hardware Description Language. Technical report, Nordic Innovation Centre, 2005.

[TS11] TÜV-SÜD. TÜV-SÜD, 2011. See `http://www.tuev-sued.de/`.

[Wei05] T. Weilkiens. Die Rolle des Systems-Engineerings. *Objekt-Spektrum*, 3:28–29, 2005. Also available as `http://www.sigs.de/publications/os/2005/03/weilkiens_OS_03_05.pdf`.

[ZAMY10] B. Zhanyuan, X. Aidong, L. Mingzhe, and S. Yan. A Novel Comparator with Hamming Code Correction for Safety Programmable Logic Controller. In *Proc. of the IEEE/Computational Problem-Solving*, pages 410–412. IEEE, December 2010.

[ZLQ10] H. Zhang, W. Li, and J. Qin. Model-based Functional Safety Analysis Method for Automotive Embedded System Application. In *Proc. of the International Conference on Intelligent Control and Information Processing*, 2010.