# A single channel, fail-safe microcontroller to simplify SIL3 safety architectures in automotive applications

## Ein einkanaliger, ausfallsicherer Mikrocontroller zur Vereinfachung von SIL3 Architekturen im Fahrzeug

Dr. **M. Baumeister**, Dipl.-Ing. **P. Fuhrmann**, Philips, Aachen;
Dr. **R. Mariani**, Yogitech, Pisa

**Kurzfassung**

Automobile Anwendungen müssen in zunehmendem Maße die Sicherheitsanforderungen der IEC 61508 erfüllen, ohne daß Komplexität und Kosten überhand nehmen. Hierzu stellen wir einen Ansatz vor, der Wissen über die internen Strukturen einer MCU ausnutzt, um zielgerichtet Tester und Überwacher in die MCU zu integrieren, der dadurch für verschiedene Anwendungen ein fail-safe Verhalten in einem einkanaligen System erreicht und dabei weniger Overhead verursacht als andere Ansätze.

**Abstract**

Automotive applications are increasingly influenced by the safety requirements of IEC 61508. Yet their complexity and costs should not increase. To support this we propose an approach for constructing a single-channel fail-safe microcontroller (MCU) able to support a variety of safety functions. Exploiting knowledge on the internal structures of the MCU allows integrating optimized fault supervisors which provide the required safety integrity, resulting in a fail-safe behavior with less overhead than other approaches.

## 1 Introduction

Upcoming innovations in the area of active safety and driver assistance systems as well as more mature applications in the area of passive protection systems ask for silicon solutions facilitating the implementation of the respective safety function in the most efficient way. At the same time, the progressive use of deep submicron technologies results in a new population of faults and failure modes, making such safety functions critical in terms of their integrity. A common way to tackle this challenge is to develop problem-specific architectures to implement the specific safety functions and to guarantee the required safety integrity level of the MCU by a ‘black box’ approach, where MCU components are either (partially) replicated (e.g. [1]) or continuously tested via its external interfaces (e.g. [2]).

This contradicts today’s platform architecture approach: Application specific safety architectures and safety integrity measures are costly and difficult to maintain during design time changes and increase product certification costs. As solution, we present a single-channel control architecture for MCUs that maintains the IEC61508 requirements needed for a SIL3 fail-safe system. In the following chapter we explain the methodology used to specify and design the MCU safety integrity architecture. We describe the basic fault used to guarantee the required safety integrity level for the relevant MCU blocks supervisors in chapter 3. Chapter 4 shows how the proposed MCU architecture is able to support different safety functions. Chapter 5 summarizes the FMEA results which show the applicability of such an MCU for SIL3 applications and chapter 6 compares its required resources with other approaches.

## 2 From black to white box approach: the fRMethodology

IEC 61508 [3] is a norm becoming popular in automotive. Thus achieving Safety Integrity Level (SIL) 3 is becoming mandatory for many automotive applications. The SIL rating is based mainly on the value of the safe failure fraction (SFF, ratio of the average rate of safe failures plus dangerous detected failures to the total average failure rate) that has to be equal or greater than 99%. Moreover, for integrated circuits (IC), the factor $\beta_{ASIC/IC}$ (quantifying the probability of common cause failures) has to be equal or lower than 25%. The foundation to compute these two metrics is the failure mode and effects analysis (FMEA). However, the number of components inside an integrated circuit is so high that system-oriented FMEA methods are practically inapplicable. As a consequence, the IC is often considered as a 'black box'. This means that the analysis is driven without knowledge of specific failure modes of the IC often resulting in duplicated or triplicated systems to achieve dependability. To break these barriers, it is mandatory to analyze the integrated circuit as a 'white box'.

The technology applied in this paper (called fRMethodology, one piece of faultRobust technology [4]) is a method for performing and validating the FMEA of any integrated circuit. In summary, the proposed method includes a first stage in which an automatic tool extracts information from the integrated circuit description by partitioning it into "sensitive zones" [5]. In a second stage, the extracted information and the optional profile are used to fill an FMEA database. This database is completed by entering information related both to possible applications and failure modes according to the IEC 61508 norm. At the end of this step, failure rates of the sensitive zones are computed and ranked according to criticality. Results collected from the FMEA database include specific indices required by IEC 61508 such as the SFF and eventually the SIL. We call the two stages just described "fRFMEA".

In a third stage, the FMEA database generated above – in particular the information related to the application provided by the system engineer – is validated using the given circuit workload by using a mix of fault injection and fault simulation (this mix is called fRFaultInjector). The results ‘measured’ from this simulation are stored in the database and compared with the computed FMEA results of the previous step. In case of a mismatch, indications are given in order to correct the previous FMEA preparation stage.

When fRMethodology is applied to a complete System on Chip (SoC) or Micro Controller Unit (MCU), a hierarchical approach is used: the SoC or MCU is divided into sub-systems and the fRFMEA is performed at different abstraction levels. Results of fRFMEA analysis of each sub-system are combined in a top-level fRFMEA giving the final results for the whole MCU in terms of Safe Failure Fraction and Diagnostic Coverage.

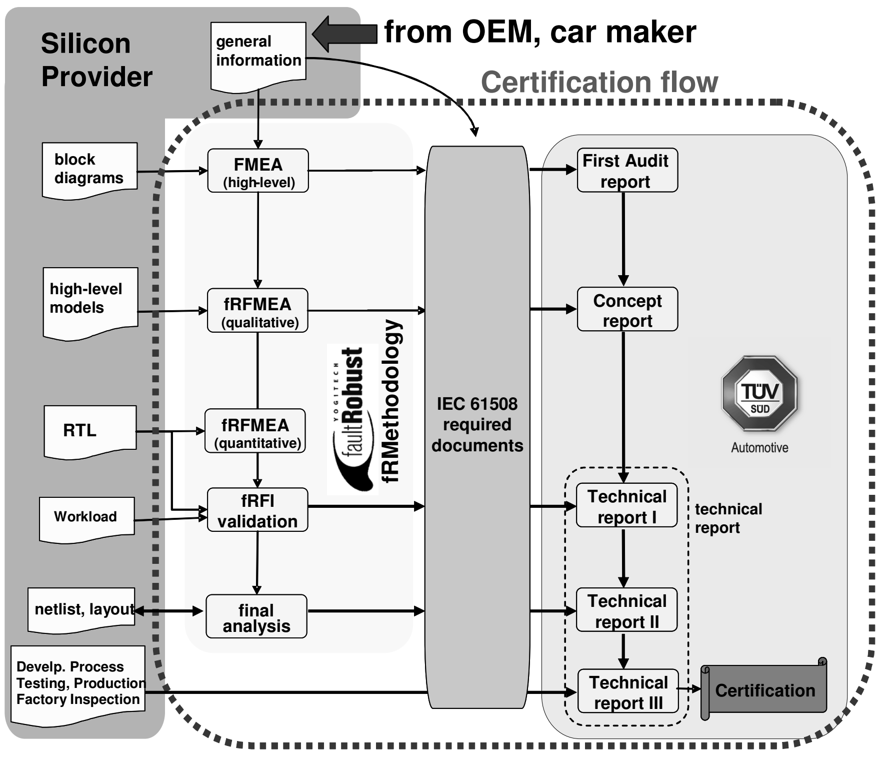
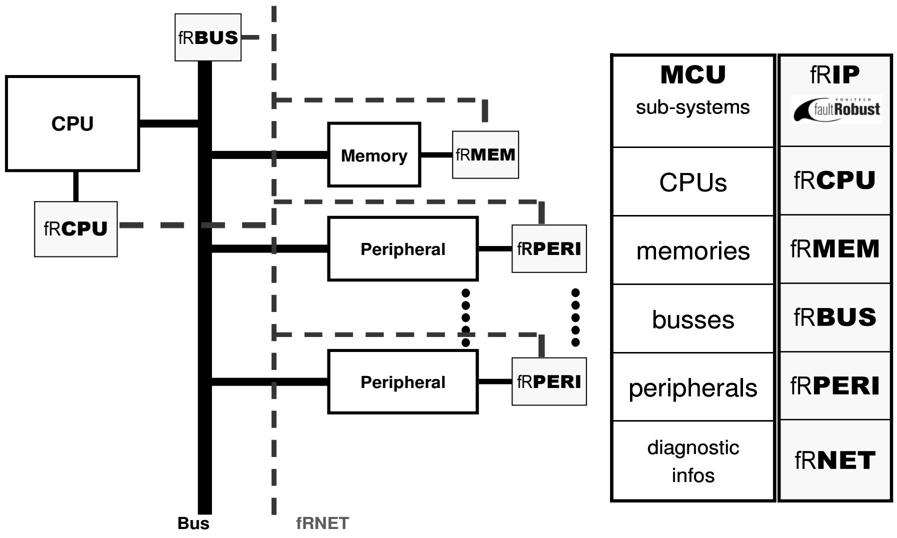


**Figure 1: The fRMethodology flow and the related MCU certification steps**

As shown in Figure 1 the analysis is performed at different abstraction levels, starting from a high-level view down to the final circuit representation. Documents are provided from the Silicon Provider (left side) to the Certification Body (such as TÜV SÜD, right side) through the proposed methodology flow (by YOGITECH, center). This is done at different milestones: initial audit, concept assessment, and the final assessment leading to certification stepping through the different MCU development stages. Assessment results determine changes and adaptation of the Silicon Provider documents, such as the Safety Manual defining the Conditions of Use of the MCU described in section 4.2.

## 3 A library of optimized fault supervisors: the fRIPs

With the information provided by fRMethodology, it is possible to design a library of optimized HW circuits (called fault supervisors) to guarantee the required safety integrity level of the MCU sub-systems used by the safety functions. These supervisors handle errors or failures either passively (detecting them) or actively (correcting them). The blocks are not intrusive and they do not require any modification of the logic circuit that they supervise. A typical infrastructure is composed of a “main” fault supervisor for the CPU (called fRCPU) and a set of “remote” supervisors (Figure 2), each one for a specified region of the system, such as the memory (fRMEM), the bus (fRBUS) and the peripheral sub-systems (fRPERIs). All fault supervisors are connected together through an on-chip “robust net” (fRNET).



| MCU sub-systems | fRIP faultRobust |
|---|---|
| CPUs | fRCPU |
| memories | fRMEM |
| busses | fRBUS |
| peripherals | fRPERI |
| diagnostic infos | fRNET |

**Figure 2: A microcontroller including the fault robust supervisors fRIPs.**

The **fRMEM** supervisor is a configurable and re-usable IP for protection of memory sub-systems [6]. It is basically an ECC-based memory protection supervisor but extended with measures to fulfill the requirements of IEC 61508, e.g. a self-checking architecture. Optionally it provides optimizations to improve access time (“fast-track”) and protection over time (“scrubbing” of developed errors). Due to the ability to store the ECC codes separately from the data, memory overhead can be reduced by providing different protection levels for different memory pages. For multi bus master architectures also a “distributed MPU” can be integrated. The fRMEM supervisor has been certified by TÜV SÜD.

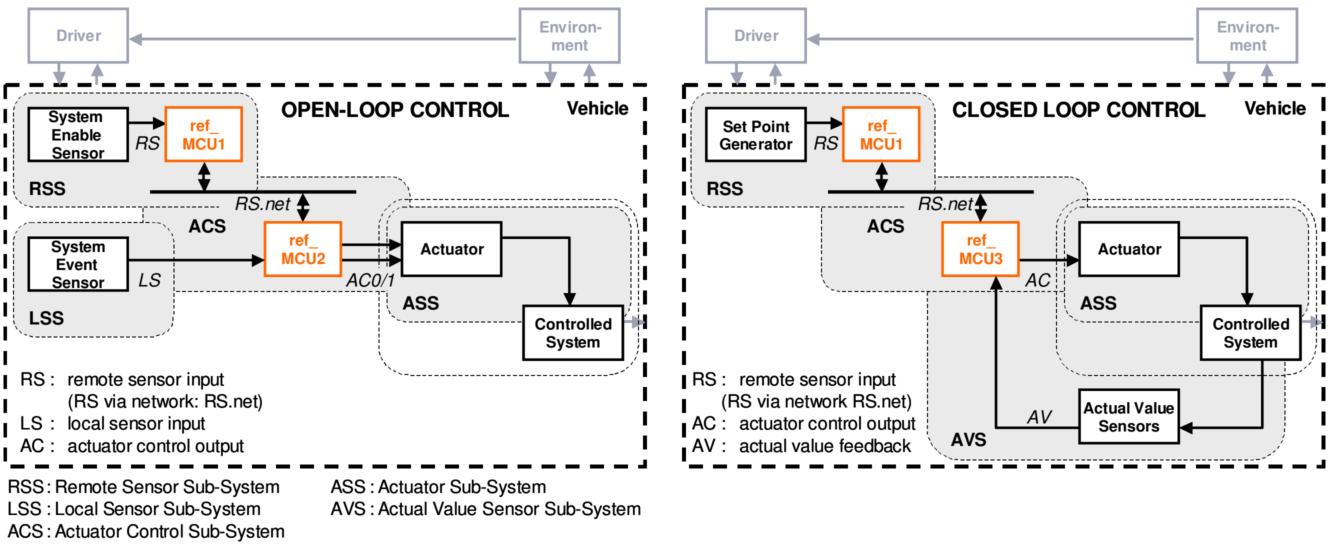
The **fRCPU** IP is designed taking full benefit of the previously described ‘white-box’ approach. Starting from a detailed fRFMEA analysis of the target CPU sensitive zones are selected which provide a SFF ≥ 99% at minimum cost. fRCPU is composed of sub units which monitor the CPU and model its behavior on an abstract level. It uses a set of independent checkers tailored to detect deviations of the CPU behavior from this model. fRCPU also includes a Coverage Monitor Unit providing run-time information on whether current usage conforms to the FMEA assumptions thus ensuring that all SFF results still hold. fRCPU does not come as a replication of the CPU, but is architecturally and functionally diverse and thus strongly reduces the $\beta_{ASIC}$ as required by IEC 61508. Also, as fRCPU implements a hardware-centric approach, most diagnostic is implemented in hardware and not in software as in approaches such as [2] or [8] and thus CPU performance is not impacted. fRCPU has been assessed by TÜV SÜD: a concept report has been delivered stating that the proposed measures can be able to fulfill the safety integrity level SIL3.

Bus supervisors (**fRBUS**) consist of decoders, arbiters, and checkers monitoring sources and sinks of the bus interconnect and controlling data integrity and routing. Peripheral supervisors (**fRPERI**) implement a "hardware verification component", i.e. a block where a subset of the protocol's checks and assertions are used to verify that a given interface and protocol are still implemented by the hardware. fRCPU, fRMEM, fRBUS and fRPERI communicate with system-level supervisors through a dedicated interconnect (**fRNET**) based on an on-chip robust protocol that guarantees that information is transferred without errors between the different diagnostic units and to a system-level supervisor such as the one described in section 4.1. This dedicated interconnect assures that safety-related information travel in a separate and parallel way with respect to mission data as required by IEC 61508.

## 4 The fail-safe MCU platform and its application in different system concepts

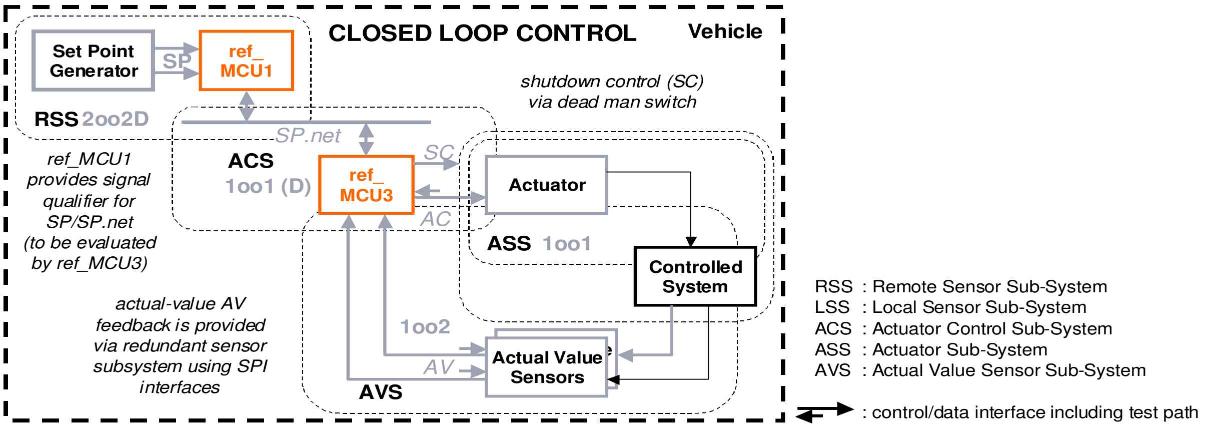
To apply the approach described above in a general purpose, fail-safe MCU applicable to a variety of safety functions, one first needs to understand the possible safety mechanisms and application constraints involved in these safety functions. Limiting ourselves to fail-safe shutdown applications, we distinguish two general control architectures of such applications: open loop control (e.g. electronic steering column lock – ESCL – and airbag systems) and closed loop control (e.g. vehicle dynamics control and other active assistive functions like EPS and ESP). We abstractly model their safety-related functionality as a distributed control system consisting of sensors, actuators, communication and microcontrollers (see Figure 3). Obviously, in these system architectures the microcontrollers (called ref_MCU) serve different application-level functions. We abstract these into three general functions: ref_MCU1 – remote sensor processing (RSS), ref_MCU2 – open-loop control processing (ACS), and ref_MCU3 – closed-loop control processing (ACS). A detailed hazard analysis together with the safe state definition for each of the considered system classes leads to safety integrity measures necessary to achieve a safe distributed control system. A possible realization for the closed loop architecture is presented in Figure 4 where gray parts indicate additional or enhanced components or communication with respect to the unprotected architecture of Figure 3.

As can be seen the ref_MCU can serve as control unit of a fail safe subsystem employing redundancy (e.g. in the RSS) although it itself is only classified as 1oo1(D). This is possible due to the presented single-channel fail-safe microcontroller being capable to integrate the independent shutdown control decision logic and to realize an independent safing or shutdown control path. This has been achieved by strictly applying fRIPs as presented in section 3 combined with a suitable and small set of additional HW- and SW-based safety integrity functions as described below.



Driver
Environ-ment
OPEN-LOOP CONTROL
Vehicle
System Enable Sensor
RS
ref_ MCU1
RSS
RS.net
ACS
ref_ MCU2
System Event Sensor
LS
LSS
AC0/1
Actuator
ASS
Controlled System
RS : remote sensor input (RS via network: RS.net)
LS : local sensor input
AC : actuator control output
Driver
Environ-ment
CLOSED LOOP CONTROL
Vehicle
Set Point Generator
RS
ref_ MCU1
RSS
RS.net
ACS
ref_ MCU3
AC
Actuator
ASS
Controlled System
AV
Actual Value Sensors
AVS
RS : remote sensor input (RS via network RS.net)
AC : actuator control output
AV : actual value feedback
RSS : Remote Sensor Sub-System
LSS : Local Sensor Sub-System
ACS : Actuator Control Sub-System
ASS : Actuator Sub-System
AVS : Actual Value Sensor Sub-System

**Figure 3: Simplified Distributed Control Systems – Open/Closed-Loop Control Systems**



CLOSED LOOP CONTROL
Vehicle
Set Point Generator
SP
ref_ MCU1
RSS 2oo2D
SP.net
shutdown control (SC) via dead man switch
ref_MCU1 provides signal qualifier for SP/SP.net (to be evaluated by ref_MCU3)
ACS 1oo1 (D)
ref_ MCU3
SC
AC
Actuator
ASS 1oo1
Controlled System
1oo2
actual-value AV feedback is provided via redundant sensor subsystem using SPI interfaces
AV
Actual Value Sensors
AVS
RSS : Remote Sensor Sub-System
LSS : Local Sensor Sub-System
ACS : Actuator Control Sub-System
ASS : Actuator Sub-System
AVS : Actual Value Sensor Sub-System
: control/data interface including test path

**Figure 4: Closed Loop Distributed Control Systems including safety integrity measures**

## 4.1 HW and SW -realized safety integrity functions

The MCU must be able to execute the various safety functions implied by the three different usages scenarios of the MCU shown above according to the specific application needs. In a one channel MCU this requires special measures since a pure MCU by itself is not able to implement and guarantee the safety integrity required for SIL3, i.e. to detect faults occurring in the microcontroller with the proper DC/SFF and to react to control the failures stemming from these faults as required for an integrated shutdown control path.

For this reason the safety MCU platform makes use of the following HW safety integrity measures which result in the concept for a safety integrity microarchitecture that has been successfully reviewed and approved by TÜV-SÜD (s. section 5):

- For the processing sub-system
  - faultRobust IPs (fRIPs) is used as presented in section 3;
- For peripheral interfaces being part of the safety-related system, e.g. ADC and GPIO
  - redundant HW is provided to enable their use in SIL3 applications.
- For the power, reset and clocking architectures
  - measures such as proper power organization, a safe clocking strategy and a proper reset architecture are applied. These strategies detect wide common-cause effects, e.g. power spikes and reset and clock faults.
- For the interactions with the communication sub-system and the external world:
  - a distributed MPU protects memory sub-systems against HW/SW faults caused by the association of the processing sub-system with further bus masters
  - a HW "signal qualification" function is supported, to generate, transfer, receive and evaluate the diagnostic information from one side to the other of the distributed control system (e.g. from ref_MCU1 to ref_MCU2).

Furthermore a system supervisor unit (SSU) is integrated to fulfill the requirement for failure control, i.e. the provision of an independent shutdown control path in case a major fault disables the processing sub-system and all its executed safety functions.

Our approach supports as well the usage of software safety integrity functions:

First, standard methods like comparing two sensor values with each other or providing end-to-end protection over the communication link using CRCs and packet counters can continue to be used without a change. This allows for an easy migration of existing safety concepts.

Second, as the processing subsystem is now basically self-checking no higher-layer test methods using e.g. challenge-response strategies (such as used in the e-gas architecture [2]) are necessary to detect faults in the working environment of the software safety functions. This reduces the necessary software overhead as well as the processing one.

Third, it is possible to propagate error indications created by the fRIPs to the software safety functions to inform them in more detail about the error cause. Using this information the software can try to either resolve the problem, e.g. by resetting specific subcomponents, or work around it, e.g. by using other, possibly less exact means to reach a comparable result. Thus our usage of the white-box approach allows software to work around occurring errors and thus to possibly increase the availability of the system.

### 4.2 Conditions of use

The fail-safe MCU is only part of the safety-relevant system. Therefore conditions exist how the MCU must be employed within the system and what behavior is expected from the system. This is generally called “Conditions of use” and is included in the Safety Manual. There are conditions implied by IEC 61508 for any such system; for example, that the correct functioning of the shutdown path must be tested. MCU-‘specific’ conditions influence how the applications running on the MCU have to be set up and react and how the surrounding system must react. Examples of the former are that applications shall enter a safe state within the system safety time when required safety-relevant data does not arrive, that applications employing an ADC must periodically check that the ADC reference voltage is measured correctly, and that the embedded SPI interfaces shall be used only for one-to-one connections unless higher-layer safety measures are provided. Examples for system conditions are predefined reactions of the system if the pins of the MCU go into high impedance mode, integration of an external capacitor into the circuitry reducing over-voltage spikes, and the expectation of a power-on reset at least every 24 hours.

## 5 Safety analysis results

The MCU concept has been assessed by TÜV SÜD with the following main results: The MCU is able to support open and closed loop automotive applications such as described in section 4. The immediate failure detection of the fRIPs used in the processing sub-system and the I/O redundancy of some of the I/Os reduces error detection latency and thus puts fewer constraints to the process safety time than other single-channel approaches.

Regarding common cause failures, the $\beta_{ASIC/IC}$ is below the required threshold of 25% due to the intrinsic diversity of fRIPs with respect to the sub-system they supervise. Contrary to dual core architectures, no additional measures as required by IEC 61508 2nd edition (part 2 annex E) for ICs have to be considered. Therefore this MCU does not require implementing guard-rings, temperature sensors and so forth. Furthermore, no external watchdog is necessary to supervise the MCU since it performs intrinsic clock supervision. Concerning parts of the MCU in which redundancy is used (such for ADC, GPIO and similar ones), conditions of use are defined in order to guarantee the avoidance of common cause failures such as distance between blocks/lines.

The concept analysis of the MCU has shown that the proposed application of fault control and fault avoidance measures can be suitable to fulfill the desired risk level SIL 3 in accordance with IEC 61508. The set of necessary conditions of use is small and easy to handle during system integration.

## 6 Comparison with different architectures

We compare our approach with two main competitors: the dual-CPU lock-step architecture and the asymmetric CPU checking architecture.

The **dual-CPU lock-step** architecture (e.g. [7]) is unable to reach SIL3 without additional measures. This is caused by the potential for common cause failures affecting both CPUs which cannot be detected by the compare unit. Therefore, the dual-CPU lock-step architecture has a $\beta_{ASIC}$ greater than 50%. HW measures required to lower the $\beta_{ASIC}$ are listed in IEC 61508. They include the use of an external watchdog, diverse synthesis for the CPU checker, specific layouting requirements (e.g. using guard-rings ≥ 100µm and a potential ring), and temperature sensors. IEC 61508 also requires a minimum diagnostic coverage of each CPU of at least 60%. Therefore proper SW routines are needed such as CPU start-up and periodic diagnostic tests, CPU compare unit management and failure diagnostic and control.

Our proposed architecture on the other hand requires no additional measures to fulfill IEC 61508 thanks to the intrinsic diversity of the fRIPs and due to the detailed analysis done with fRFMEA as described in section 2 and 3.

The above measures increase the lock-step architecture's overhead over the one caused by replication alone. This can be seen in Table 1 which summarizes the costs of the approaches based on a reference design of a ARM968ES dual-CPU lock step architecture. The values represent the delta cost for the safety integrity measures, i.e. the cost of the architecture with safety integrity measures minus the cost of the architecture without them. Moreover, costs are normalized to the costs of the pure dual-CPU lock-step reference as shown in the grey column. These values were determined by extracting information from the reference design (e.g. gate counts after synthesis, memory area, power computation) and by estimating SW overheads based on commonly used diagnostic routines. It can be seen that when focusing on the CPUs the fR approach has around one sixth of the delta costs of a SIL 3 compliant dual-CPU lock-step architecture. This is mainly caused by fR not using a simple replication of functionality but instead employing white box verification of the correct functioning of the CPU.

| | | **SIL3** | |
|---|---|---|---|
| | **Dual CPU Lock Step $\beta_{asic/IC}$>25%** | **Dual CPU Lock Step $\beta_{asic/IC}$≤25%** | **faultRobust (fRCPU)** |
| **Area overhead** | 1 | 1.97 | 0.31 |
| **SW overhead** | 1 | 2.22 | 0.45 |
| **Power overhead** | 1 | 1.94 | 0.31 |
| **Detection latency** | 1 | 3 | <1 |
| **Performance overhead** | 1 | 2.22 | 0.08 |

**Table 1: Comparison of approaches without memories**

Including large memories in the analysis dilutes the advantage of the fR approach as memory protection includes a constant overhead e.g. for ECC storage. An obvious question is how significant this diluting effect is. For a typical medium-high end automotive MCU with 512Kb flash, 48Kb SRAM, 32Kb shared TCM (Tightly Coupled Memories) and 0.25Mgates of extra logic, the saving of fRCPU/fRMEM relative to dual-CPU lock-step is still around 25%. Savings grow if TCM memories are not shared between the lock-step CPUs. This benchmark also does not yet include the optimized supervision of the buses and peripherals for failures. With a simple simulation, for the typical medium-high end automotive MCU described before and considering 70% of the extra logic to be protected with HW safety integrity measures, the saving of fRCPU/fRMEM plus fRBUS/fRPERI with respect to dual-CPU lock-step plus bus/peripheral redundancy rapidly grows to 35%.

The **asymmetric CPU checking architecture** (as used in [2] and [8]) is composed of two diverse CPU cores (or a CPU core and a DSP, or a CPU core and an external smaller CPU). Typically asymmetric checking is implemented through a ‘challenge-response’ mechanism (C&R), i.e. the second CPU core send “challenges” to the main CPU and checks if the responses are in line with expected values. In other implementations, the second CPU core interrupts the execution of the main CPU core to execute some kinds of diagnostic tests.

Main drawbacks of such an architecture compared to the fR approach are the following:

- The C&R mechanism by definition cannot achieve full **coverage** of transient faults, unless they are present during the period of time in which the C&R operation or the diagnostic tests are running. So SIL3 is very difficult to be achieved.
- The **HW overhead** is significantly greater than the one to be paid for fR approach. In most of the cases the second CPU has to be powerful enough to generate sophisticated “challenges” (otherwise coverage will decrease) and it has to be provided with a dedicated memory sub-system, bringing high integration costs.
- The diagnostic coverage is clearly proportional to the **SW overhead** caused by the C&R. Moreover, this architecture is often supported by some additional SW diversity (such as variable mirroring) to cover remaining unprotected sub-systems. For complex applications, this SW overhead can lead to an unacceptable increase of flash memory.
- Safe Failure Fraction and diagnostic coverage are directly proportional to how often and for how long C&R or diagnostic tests are performed. The same holds for diagnostic capability. This significant **performance overhead** can make this architecture unsuitable for small microcontrollers with no additional performance headroom.

## 7 Conclusions

We have shown how the combination of a ‘white-box’ risk analysis process, of fault supervisor components for CPU, memory, and bus, of specific hardware measures for critical support components, and of some conditions of use result in a fail-safe one-channel SIL3 capable MCU able to support a variety of safety functions. As demonstrated by the performed cost benchmark this solution compares favorably with other approaches with regards to HW and SW overhead. Since the protected processing system of such a fail-safe microcontroller allows an application independent analysis of the diagnostic coverage capabilities and of the safe failure fraction according to IEC 61508, the safety concept developer gains additional degrees of freedom for the E/E control architecture realization. Furthermore, the overall validation and certification of the safety case is drastically simplified. Moreover, the provided enhanced diagnostic capability will allow system makers and SW application designers to increase availability and to exploit new ideas such as dynamic reconfiguration of HW resources in case of failures.

## 8 References

[1] M. Baleani, et al. "Fault-Tolerant Platforms for Automotive Safety-Critical Applications”. In International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES'03). San Jose, October 2003

[2] Arbeitskreis EGAS: Standardisiertes E-Gas-Überwachungskonzept für Motorsteuerungen von Otto- und Dieselmotoren. Report Version 2.0. Verband der Automobilindustrie, May 2005.

[3] CEI International Standard IEC 61508, 1998-2000.

[4] faultRobust technology, www.fr.yogitech.com

[5] R. Mariani, G. Boschi, and F. Colucci, “Using an innovative SoC-level FMEA methodology to design in compliance with IEC61508”, Proceedings of DATE 2007, April 2007, Nice, France

[6] R. Mariani, F. Colucci, and P. Fuhrmann, Safety integrity of memory sub-systems in automotive microcontrollers, SAE 2007 World Congress, 2007-01-1494

[7] T. Fruehling, “Delphi Secured Microcontroller Architecture”, SAE Technical Paper, 2000-01-1052

[8] S. Brewerton, R. Schneider, and D. Eberhard, „Implementation of a Basic Single-Microcontroller Monitoring Concept for Safety Critical Systems on a Dual Core Microcontroller“, SAE 2007 World Congress, 2007-01-1486