CHALMERS
UNIVERSITY OF TECHNOLOGY

An ISO 13849 compatible Emergency Stop System on a high-performance system-on-chip hardware platform

Master's thesis in Embedded Electronic System Design

Ruxuan Wen, Wenbo Tan

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Master's thesis 2026

An ISO 13849 compatible Emergency Stop System on a high-performance system-on-chip hardware platform

Ruxuan Wen, Wenbo Tan

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2026

An ISO 13849 compatible Emergency Stop System on a high-performance system-on-chip hardware platform
Ruxuan Wen, Wenbo Tan

Supervisor: Lars Svensson, Department of Microtechnology and Nanoscience
Company advisor: Jonas Martinsson, Subramanian Murali Ram, Volvo Group
Examiner: Per Larsson-Edefors, Department of Microtechnology and Nanoscience

Master's Thesis 2026
Department of Microtechnology and Nanoscience
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

An ISO 13849 compatible Emergency Stop System on a high-performance system-on-chip hardware platform
Ruxuan Wen, Wenbo Tan
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

This thesis presents the design, implementation, and evaluation of a heterogeneous dual-channel Emergency Stop safety function on a Xilinx ZCU104 evaluation board. Channel A is deployed on the Cortex-R5 in the Processing System and Channel B on a MicroBlaze soft processor in the Programmable Logic, introducing diversity through processor architecture, memory organisation, and timing model. Both channels independently sample the E-Stop inputs, execute a four-state safety state machine, and exchange safety words over a mailbox for bidirectional cross monitoring.

Diagnostic coverage is estimated using the simplified ISO 13849-1 Annex E method. Each channel implements the three measures required for the logic subsystem: dual-copy variable memory with per-tick byte comparison, periodic CRC signature verification of invariable memory, and processing-unit self tests. Independent watchdog supervision is provided on each channel. A UART fault injection interface enables deterministic validation of all diagnostic fault paths.

The overall diagnostic coverage is 96% for Channel A and 93% for Channel B, both in the medium range. The common cause failure score is estimated at 80, exceeding the Annex F minimum required for Category 3. An $\mathrm{MTTF_D}$ analysis yields a capped value of 100 years (High tier). With Category 3 architecture, high $\mathrm{MTTF_D}$, medium $\mathrm{DC_{avg}}$, and a passing CCF score, both channels achieve Performance Level d.

Keywords: ISO 13849-1, Emergency Stop, diagnostic coverage, Performance Level

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Lars Svensson, for his guidance throughout this thesis and for his valuable suggestions that helped us further polish and improve the report. We also thank our examiner, Per Larsson-Edefors, for his input on the thesis structure and methodology.

We are grateful to Volvo Group for supporting this thesis project. In particular, we would like to thank Jonas Martinsson and Subramanian Murali Ram for their continuous support in our day-to-day stand-up meetings, for their patience, and for the many times they helped us regain direction when the work felt uncertain. Although we were not co-located during the project, their guidance and support consistently bridged the distance and played an important role in moving the work forward.

We would also like to thank our colleagues at the Gothenburg office for their kindness, encouragement, and support during the project.

Wenbo Tan & Ruxuan Wen, Gothenburg, June 2026

Contents

List of Abbreviations

AGV	Automated Guided Vehicle
AV	Autonomous Vehicle
AVC	Autonomous Vehicle Control
CAN	Controller Area Network
CCF	Common Cause Failures
CRC	Cyclic Redundancy Check
DC	Diagnostic Coverage
DCavg	Average Diagnostic Coverage
DoS	Denial of Service
E-Stop	Emergency Stop
FPGA	Field-programmable gate array
FTA	Fault Tree Analysis
GPIO	General Purpose Input Output
HIL	Hardware in the Loop
ISO	International Organization for Standardization
$MTTF_D$	Mean Time To Dangerous Failure
NC	Normally Closed
PFH	Average Frequency of Dangerous Failure per Hour
PL	Performance Level
PLr	Required Performance Level
PS	Processing System
SCM	Safety Control Module
SIL	Safety Integrity Level
SoC	System on Chip
SRP/CS	Safety-related parts of a control system
SRS	Safety Requirements Specification
STAMP	Systems Theoretic Accident Model and Processes
STPA	System Theoretic Process Analysis

1

Introduction

Transportation serves as the foundation of modern society. Currently, automation is driving a revolution at enhancing safety and operational efficiency. Recently, Autonomous Vehicles (AVs) have become a focal point for both academia and industry. Controlled environments, such as ports, mines, and logistics hubs, offer a high degree of predictability. Therefore, they are considered ideal for the early deployment of AV technology [1]. Despite their potential to optimize traffic safety, AVs introduce complex security and safety challenges. Failures or damage within system components can trigger a chain reaction, disrupting intra-vehicle communication networks. Such disruptions may lead to decision-making biases in the onboard computer, potentially resulting in accidents [2]. Additionally, as vehicle functions become increasingly integrated and road environments grow more complex, the risk of safety system failure rises accordingly [3]. Consequently, it is imperative to develop and implement robust countermeasures that address functional safety. These measures are essential to effectively prevent and mitigate potential safety losses in autonomous operations.

In functional safety engineering for machinery, risk reduction starts with identifying hazards and determining how these hazards can lead to harm. When hazards cannot be fully eliminated by design or by purely mechanical protective measures, risk reduction is commonly realized through safety functions implemented by the control system. A safety function is a function implemented by a Safety-Related Part of a Control System (SRP/CS), which is needed to achieve or maintain a safe state for the machine with respect to a specific hazardous event [4]. The Emergency Stop (E-Stop) system is treated as a safety function because its failure can result in an immediate increase of risk.

The E-Stop system is an essential fail-safe component in autonomous vehicle architecture. Its main benefit is that it provides a separate way to stop the vehicle without relying on the normal control algorithms. This mechanism ensures the system can be returned to a predefined ”safe state” [5]. By doing so, it effectively prevents or minimizes risks that could lead to injuries or property damage [6]. Although advanced perception and control algorithms can avoid most collisions, they may not be reliable as a fail-safe if they are part of the normal control system. Therefore, a separate, independent safety mechanism is needed to keep the vehicle safe when unexpected errors occur [7][8]. In safety kernel designs, the E-Stop serves as the highest-priority controller. This allows it to override all other motion commands to ensure a forced stop during extreme situations [6].

For the confined area application context considered in this thesis, ISO 13849 pro-

vides the relevant framework for quantifying and validating the reliability of safety functions. This standard provides standardized guidance for the design, integration, and validation of SRP/CS [4]. It transforms complex safety performance into technical indicators that are both predictable and verifiable. The core concept of this standard is the Performance Level. The Performance Level, or PL for short, is used to evaluate the ability of a system to perform safety functions under defined conditions. In the current fields of automotive and industrial automation, ISO 13849 offers a clear and rigorous framework. It is used to assess both hardware and software. Therefore, we apply ISO 13849 performance level to quantify and verify the reliability of the E-Stop safety function.

The core objective of this thesis is to implement a prototype-level, ISO 13849-oriented E-Stop system on a heterogeneous embedded evaluation platform. The selected platform supports the architectural investigation of redundant safety channels, cross-monitoring, and run-time diagnostics within a proof-of-concept scope. It is therefore used to validate the feasibility of a deployment-oriented stop-function concept, rather than to represent a finished product platform. The primary challenge of this implementation is to ensure that safety functions remain undisturbed while performing high-performance computing and monitoring tasks.

The E-Stop function is designed to achieve Performance Level d (PLd) and Category 3 requirements, as described later in Section 2.5. Typically, to determine the Performance Level, the evaluation process follows several structured steps. A risk assessment is performed to define the necessary safety requirements. This is followed by defining the system architecture, which provides the structural basis for the safety function. Additionally, specific component parameters must be determined, including the Mean Time to Dangerous Failure ($MTTF_D$) and Diagnostic Coverage (DC). The system is checked for Common Cause Failures (CCF) to ensure that external factors do not compromise redundant paths. Consequently, the calculated results are compared with standard tables to verify whether the achieved Performance Level satisfies the requirements of the ISO 13849 standard [9].

1.1 Related work

As autonomous driving levels advance, vehicle electronic architectures are transitioning from traditional decentralized structures to integrated domain control architectures. Traditionally, safety functions were managed by independent Safety Controllers. While these provided physical isolation, they limited the system's ability to handle complex safety logic and secure communications. Jiang et al. [10] pointed out that the Controller Area Network (CAN) protocol lacks robust security encryption, making it highly vulnerable to cyber-attacks such as replay attacks, Denial of Service (DoS), and message spoofing. Such communication vulnerabilities could lead to the malicious interception or tampering of Emergency Stop (E-Stop) signals, resulting in catastrophic functional failure. Consequently, integrating safety functions into high-performance computing platforms has become an industry trend to improve response efficiency and mitigate security risks.

The E-Stop system is the most critical component of the autonomous driving safety

architecture, serving as the final line of defense. Its reliability directly determines vehicle safety during extreme failure scenarios. To enhance the robustness of E-Stop mechanisms, extensive research has focused on optimizing E-Stop triggering and control algorithms across various automated systems. Capannini et al. [11] introduced a visibility-based algorithm to identify and halt machinery within the line of sight of an emergency button, enhancing response reliability in industrial settings. Wu et al. [12] combined computer vision with deep learning models to develop intelligent E-Stop systems for an industrial robot workstation. Regarding maneuver control, Duerr et al. [13] proposed a global optimization algorithm for arbitrary electrical failures to ensure the vehicle reaches a safe state after an E-Stop is triggered. Furthermore, for multi-robot scenarios, Ogorelysheva et al. [5] investigated collision mitigation strategies for AGV fleets to resolve group safety issues during control failures.

To address the critical nature of E-Stop functions, researchers have adopted rigorous design and analysis methods. Porras-Vázquez et al. [14] developed a new verification methodology for the ISO 13849 standard, simplifying the design process for SRP/CS. For unprecedentedly complex systems, Watanabe and Itoh [6] demonstrated the effectiveness of the STAMP/STPA safety analysis method in identifying hazards that traditional analyses often overlook. Quantitative evaluations have been applied in other fields. Kaczor et al. [15] explored SIL verification and $MTTF_d$ calculation in railway systems. Allouch et al. [16] conducted qualitative and quantitative risk assessments for unmanned aerial vehicles missions. These multi-dimensional analytical methods provide the theoretical foundation for implementing high-integrity safety functions on a single-chip platform.

Meeting the Category 3 requirements of ISO 13849 necessitates robust hardware redundancy and self-diagnostic capabilities. Bao et al. [17] studied a highly redundant safety scheme for commercial vehicle pneumatic braking systems, validating its reliability through Fault Tree Analysis (FTA). During the verification phase, real-time simulation is essential to ensure temporal correctness. Veneroso et al. [18] utilized Hardware-in-the-Loop (HIL) simulation for the preliminary design of Electronic Power Steering systems. Additionally, We et al. [19] proposed an ECU-in-the-Loop real-time simulation technique. This approach not only validates functional logic but also ensures that safety monitoring tasks satisfy real-time constraints while running in parallel with high-performance computing tasks.

1.2 Purpose and goal

The primary purpose of this thesis is to deploy an Emergency Stop safety function prototype on a heterogeneous multi-processor SoC, and to demonstrate that the implemented E-Stop system can achieve ISO 13849 Category 3 with a target PLd. Specifically, the thesis aims to achieve the following:

- **Achieve ISO 13849-1 Performance Level d compliance:** Design a dual-channel architecture intended to meet the requirements for Category 3 and Performance Level d within the prototype scope. The design shall achieve medium average diagnostic coverage, which is $90\% \leq DC_{avg} < 99\%$. The

common cause failures (CCF) score should be 65 points at least.

- **Implement Heterogeneous 1oo2 Redundancy:** Construct a robust 1oo2 (One-out-of-Two) architecture by utilizing core A as the primary channel and core B as the secondary diversity channel. This setup aims to leverage hardware diversity to protect against random failures while ensuring real-time responsiveness.
- **Validate the design:** Perform bench-level fault injection tests to empirically verify the E-stop behavior under representative fault conditions and to support the diagnostic coverage argument and overall safety-case evidence for the targeted category and performance level.

1.3 Thesis outline

Chapter 2 introduces the technical background relevant to ISO 13849-1, including SRP/CS, Performance Level, architectural categories, and key evaluation parameters such as MTTFD, DC, and CCF.

Chapter 3 describes the applied method, including the ISO 13849-1-oriented development workflow, hardware failure-rate estimation, diagnostic coverage evaluation, and validation through fault injection.

Chapter 4 describes the design and implementation of the proposed E-Stop safety function. It covers the two-channel architecture, channel allocation rationale, the four-state safety state machine, the cross-monitoring path and safety-word format, and the complete set of diagnostic measures. The chapter closes with a summary of design constraints relevant to the prototype.

Chapter 5 reports the observed behavior of the implemented prototype. It begins with an implementation status overview, then presents the state-machine behavior, Category 3 redundancy results, and the verification and validation campaign using the UART fault injection interface. Diagnostic coverage estimation, common cause failure assessment, and $MTTF_D$ analysis are then presented, together with a Performance Level summary.

Chapter 6 discusses the architectural asymmetries between the two channels, the scope and limitations of cross-monitoring, the output-stage constraints, and the diagnostic coverage compensation mechanism that allows both channels to reach the same Performance Level despite asymmetric logic diagnostics. Ethical considerations regarding data handling and the use of generative AI tools are also addressed.

Chapter 7 summarizes the contributions, lists the current limitations, and outlines directions for future work.

2

Technical Background

This chapter introduces the theoretical and standardization background required to understand the development and assessment of an emergency stop safety function implemented by a safety-related control system. The focus is on established concepts and terminology from ISO 13849-1 [4] and related functional safety principles. Certain specific hardware, tools, and implementation choices are intentionally omitted for confidentiality.

2.1 SRP/CS and safety function

In the context of machinery safety, risks are reduced through a structured process that starts with risk assessment and continues with the selection and implementation of risk reduction measures. ISO 13849-1 applies when risk reduction relies on safety functions implemented by SRP/CS. An SRP/CS is the part of a control system that performs a safety function, starting from safety-related inputs and ending at safety-related outputs.

A safety function is defined as a function of a machine, the failure of the function can result in an immediate increase of risk. Typical safety functions include safety-related stop functions and emergency stop functions, where the control system is responsible for bringing the machine to a safe state and maintaining that state until conditions for restart are met.

2.2 Performance Level

ISO 13849-1 defines the achieved performance level as a discrete measure of the ability of an SRP/CS to perform a safety function under foreseeable conditions. The Performance Level is linked to the average frequency of dangerous failure per hour. The standard provides average frequency of dangerous failure per hour (PFH) ranges for PL a through PL e, where PL e corresponds to the lowest PFH range.

In general, the achieved PL for a safety function must meet or exceed $\mathrm{PL_r}$.

2.3 Risk assessment and required performance level

ISO 13849-1 is used as part of the overall risk reduction framework, where the outcome of risk assessment determines which safety functions are required. For each

safety function, a required performance level (PL_r) is determined. PL_r expresses the safety performance needed to achieve the intended risk reduction.

A commonly used guidance method for determining PL_r is the risk graph in ISO 13849-1 Annex A [4], which considers:

- severity of injury (S),
- frequency and/or duration of exposure to the hazard (F),
- possibility of avoiding or limiting harm (P).

The result is a discrete PL_r from a (lowest) to e (highest).

2.4 Subsystems and decomposition of an SRP/CS

A safety function is often realized by multiple subsystems, typically spanning input, logic, and output elements. ISO 13849-1 allows decomposition of the SRP/CS into subsystems. A dangerous failure of any subsystem can result in loss of the overall safety function. This decomposition supports systematic evaluation and documentation, and is also used when combining subsystem performance to obtain an overall PL for the safety function.

2.5 Architectural categories

ISO 13849-1 uses categories to classify subsystem architectures with respect to fault resistance and behavior under fault conditions [4]. The categories are:

- **Category B**: applies basic safety principles, but a single fault may cause loss of the safety function.
- **Category 1**: builds on Category B by using well tried components and well tried safety principles so faults are less likely but can still cause loss of the safety function.
- **Category 2**: adds periodic testing of the functional channel so a fault may cause loss of the safety function between tests while the test is intended to reveal the fault.
- **Category 3**: uses a redundant structure so a single fault does not cause loss of the safety function while only some faults are detected and an accumulation of undetected faults can still cause loss of the safety function.
- **Category 4**: also uses redundancy and requires higher diagnostics so single faults are detected in time and the design addresses accumulated faults to prevent loss of the safety function.

The category defines the expected fault behaviour of the subsystem and constrains the achievable PL. The final PL depends on the selected category together with parameters such as $MTTF_D$ of each functional channel, DC_{avg}, and measures against common cause failures (CCF), as specified in ISO 13849-1 [4].

2.6 Key parameters: $MTTF_D$, DC, and CCF

2.6.1 Mean time to dangerous failure

$MTTF_D$ characterizes the reliability of components or channels with respect to dangerous failures. In ISO 13849-1, $MTTF_D$ is one of the key parameters used to evaluate the achieved performance level of a safety-related part of a control system. The standard defines three ranges for $MTTF_D$ per channel and provides guidance for estimation using manufacturer data, calculation methods, or field data. For interpretation within ISO 13849-1, the $MTTF_D$ of each channel is classified into three ranges, as shown in table 2.1. For components with a constant dangerous failure

Table 2.1: $MTTF_D$ Index to Time

Index	Range (time)
low	3 years $\leq MTTF_D <$ 10 years
medium	10 years $\leq MTTF_D <$ 30 years
high	30 years $\leq MTTF_D \leq$ 100 years

rate, the standard states that $MTTF_D$ is numerically equal to the reciprocal of the dangerous failure rate. Therefore, the fundamental relation between $MTTF_D$ and dangerous failure rate is

$$MTTF_D = \frac{1}{\lambda_D} \tag{2.1}$$

where λ_D represents the rate at which dangerous failures occur. In practice, failure rate data is often expressed in Failures in Time(FIT), which represents the number of failures per 10^9 operating hours. The conversion between FIT and dangerous failure rate is

$$\lambda_D = FIT \times 10^{-9}/\text{h} \tag{2.2}$$

Substituting this relation into the $MTTF_D$ definition gives

$$MTTF_D = \frac{1}{FIT \times 10^{-9}}(\text{hours}) \tag{2.3}$$

2.6.2 Diagnostic coverage

Diagnostic coverage (DC) measures the effectiveness of diagnostics as the ratio of detected dangerous failure rate to total dangerous failure rate. ISO 13849-1 defines DC levels (none, low, medium, high) and provides examples of diagnostic measures (e.g., monitoring, cross monitoring, self-tests) and indicative DC values in Annex E. For systems with multiple diagnostic measures, an average diagnostic coverage DC_{avg} can be estimated.

2.6.3 Common cause failures

Common cause failure refers to events that can cause concurrent failures of multiple channels in a redundant subsystem, potentially defeating redundancy. For categories

2, 3, and 4, ISO 13849-1 requires measures against CCF and provides a scoring-based approach in Annex F to demonstrate sufficient mitigation.

2.7 Monitoring and cross monitoring

ISO 13849-1 describes monitoring as a diagnostic measure that detects a state and compares it to an expected value. Monitoring may be implemented through plausibility checks, cyclic tests, or other mechanisms. Cross monitoring is a specific diagnostic measure that checks the plausibility of redundant signals in both channels of a redundant subsystem. These concepts are central for achieving DC in categories that rely on diagnostics (categories 2 to 4).

2.8 Safety-related stop behavior

A safety-related stop function shall, as soon as necessary after actuation, put the machine into a safe state and have priority over relevant starts and non-safety-related stops. After a stop command has been initiated by a safety function, the stop condition shall be maintained until safe conditions for restarting exist.

If indicated by risk assessment, the cancellation of the stop command may require a manual reset function. ISO 13849-1 specifies that manual reset shall be initiated by a separate manual action, shall not itself initiate a hazardous situation, and shall only enable the control system to accept a separate start command. The reset actuator placement and visibility considerations aim to prevent restarting while a person is in the hazard zone .

2.9 Systematic failures and software

ISO 13849-1 distinguishes random failures from systematic failures. Systematic failures are deterministic and can result from errors in specification, design, implementation, procedures, or inadequate consideration of environmental conditions. The standard emphasizes a systematic design approach, documentation, and appropriate verification and validation activities to control systematic failures [4].

When software is part of the SRP/CS, ISO 13849-1 provides software safety requirements and lifecycle expectations (e.g., verification and validation activities and documentation) to support the required PL. It also highlights that when safety and non-safety functions share hardware resources, it shall be demonstrated that non-safety functions cannot block or delay the required safety response under normal or fault conditions [4].

2.10 Verification and validation overview

ISO 13849-1 separates verification (confirming specified requirements have been fulfilled) from validation (confirming suitability for intended use). Validation combines

analysis and testing, and for categories 2, 3, and 4 it includes testing under fault conditions, potentially using fault injection, to demonstrate that diagnostic functions trigger appropriate fault reactions. Proper validation records and technical documentation are required to support claims about achieved PL and compliance with category requirements [4].

3
Method

This thesis project will apply the iterative safety-related parts of control systems (SRP/CS) design and assessment process described in ISO 13849-1 [4]. The methodology will be structured around establishing a requirements baseline from hazard analysis, formulating a hardware architecture, and evaluating the intended performance level using analytical calculations and fault-injection validation.

3.1 ISO 13849-1 workflow and requirements

The development process will adhere to the iterative safety lifecycle defined by ISO 13849-1. The work will be driven by the risk assessment outcomes for the addressed autonomous vehicle hazards, which will provide the basis for specifying the safety functions and their required performance level (PL_r). A requirements baseline will be established for each safety function to guide the high-level SRP/CS description. The selected architecture category will be documented in terms of functional subsystems and interfaces to support traceability. This workflow is inherently iterative and will be revisited when design gaps, such as unclear fault reactions or incomplete diagnostic argumentation, are identified during subsequent evaluation phases.

3.2 Tool-driven hardware failure-rate estimation

Evaluating the performance level will rely heavily on the mean time to dangerous failure ($MTTF_D$) per channel. For the programmable logic domain, the AMD safety manual states that random hardware fault integrity is the responsibility of the system integrator. Rather than assigning a generic device-level failure rate to the entire integrated circuit, this methodology will utilize a tool-driven approach to estimate the specific failure contribution of the resources allocated to the safety-related design hierarchy.

The AMD failure-rate analysis flow within the Vivado environment will be employed alongside standard failure modes, effects, and diagnostic analysis (FMEDA) tools to enable post-implementation estimation. Automated Tcl scripts will be executed after place-and-route to extract failure-rate results based on the actual physical resource utilization of the FPGA logic. The total permanent and transient failure rates of the safety-related logic will subsequently be derived from these outputs and will serve as the direct quantitative input for the $MTTF_D$ calculation.

3.3 Diagnostic coverage and evaluation

The main analytical effort will focus on establishing and justifying the average diagnostic coverage (DC_{avg}). Implemented diagnostic measures will be identified, mapped to the standard ISO 13849-1 diagnostic coverage guidance, and synthesized using the prescribed estimation framework. The resulting parameter set, combining the selected category, $MTTF_D$, DC_{avg}, and common cause failure (CCF) control measures, will then be used to determine the achieved performance level according to the simplified procedure framework of the standard.

3.4 Validation through fault injection

Verification and validation will be conducted to confirm that the implemented safety functions, such as the emergency stop, fulfil the safety requirements specification under foreseeable conditions. Verification will ensure that all performance assessment inputs remain consistent, traceable, and correctly applied throughout the design iterations. Validation will combine theoretical analysis with empirical testing to verify correct state transitions, the retention of the safe state, and proper manual reset behavior.

Since the targeted dual-channel architecture relies extensively on runtime diagnostics, the validation strategy will incorporate rigorous fault injection testing. This will involve a dual approach consisting of hardware-level physical disconnections on the bench to simulate external wiring failures, and software-administered fault injections via a dedicated communication interface to emulate internal processing faults. The software injection strategy will target specific diagnostic mechanisms, including memory integrity checks, processing unit health monitoring, and cross-monitoring consistency. These procedures will be designed to demonstrate that representative faults are robustly detected and trigger the intended safe-state reaction without requiring permanent physical corruption of the test hardware. Test evidence will be documented through formal records supplemented by runtime log outputs from the diagnostic mechanisms.

4

Design

This chapter describes the design and implementation of the proposed E-stop safety function. The focus is on the implemented system architecture, state-machine behaviour, cross monitoring mechanism, and diagnostic measures used in the prototype.

4.1 Design objectives

The design objective of the prototype is to implement an Emergency Stop function that transitions the system into a defined safe condition upon demand and maintains that condition until an intentional recovery sequence has been completed. In practical terms, the safe state intended by the E-Stop function is the removal of the command that would allow hazardous vehicle motion, such that the actuator side would be driven to a stop-enforcing condition. In the present prototype, this safe state is represented at the logic and indication level only.

The design follows an ISO 13849 oriented two channel concept. The purpose of the architecture is not only to realize the E-stop function itself, but also to demonstrate how redundant decision paths, cross monitoring, and run-time diagnostics can be mapped onto a heterogeneous SoC platform. The target is therefore a prototype that supports a Category 3 style safety argument through redundancy, monitoring, and fault reaction.

To realize these design objectives, the prototype was implemented on a Xilinx ZCU104 evaluation board [20]. This choice reduced implementation complexity and made it possible to focus on the safety architecture, diagnostic mechanisms, and validation methodology within the available project scope.

Another reason for selecting this platform was the availability of vendor provided safety documentation, including a safety manual, which supported the safety-oriented design and analysis work and reduced the effort required to establish a structured implementation approach. In addition, prior experience from earlier master level project work with development platforms from the same manufacturer enabled more efficient use of the toolchain and shortened the overall integration effort.

The board was therefore selected primarily for prototyping efficiency, documentation support, and practical suitability for bench level experiments, rather than as a representation of a final production platform.

4.2 Two-channel architecture

The implemented prototype consists of two redundant safety channels, denoted Channel A and Channel B. Each channel independently acquires the safety-related inputs, executes the safety state machine locally, exchanges safety words with the other channel for bidirectional cross monitoring, and drives channel-local indicator outputs. In this way, both channels maintain their own safety decision path while also supervising the consistency of the remote channel status through a dedicated communication interface.

The proposed E-stop system follows the input-logic-output decomposition defined in ISO 13849-1, as illustrated in Figure 4.1. This standard establishes that a safety function is theoretically realized through a chain starting from the initiating event at the inputs and ending at the outputs that command the safe reaction.

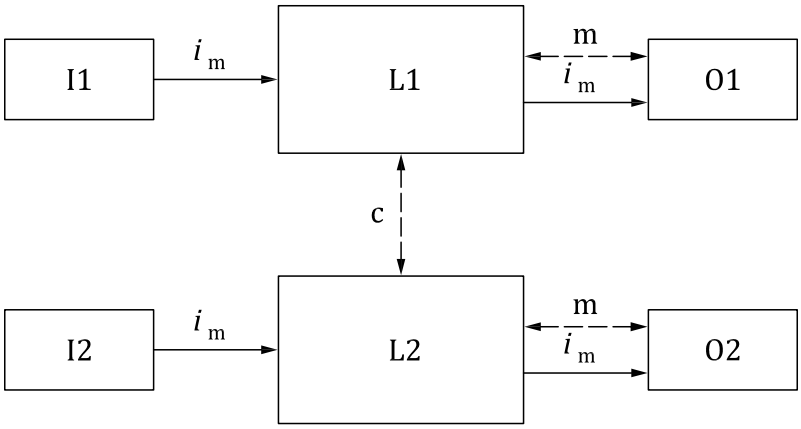

Figure 4.1: ISO 13849-1 inspired input-logic-output decomposition for a two-channel safety-related control [4].

This abstract structure is concretely realized on the targeted heterogeneous platform as the dual-channel architecture shown in Figure 4.2. In this implementation, each channel corresponds to one independent input-logic-output chain. The input subsystem captures the emergency stop and reset requests, the logic subsystem performs local safety decision making and diagnostics, and the output subsystem exposes the resulting channel-local safety indications.

The local outputs (A_out and B_out) serve as local indicators for prototype validation and behavioral observation. This means that the present implementation demonstrates redundant decision making, local fault reaction, and mutual supervision between the two channels. Therefore, the output subsystem should be interpreted as a prototype representation of channel behavior.

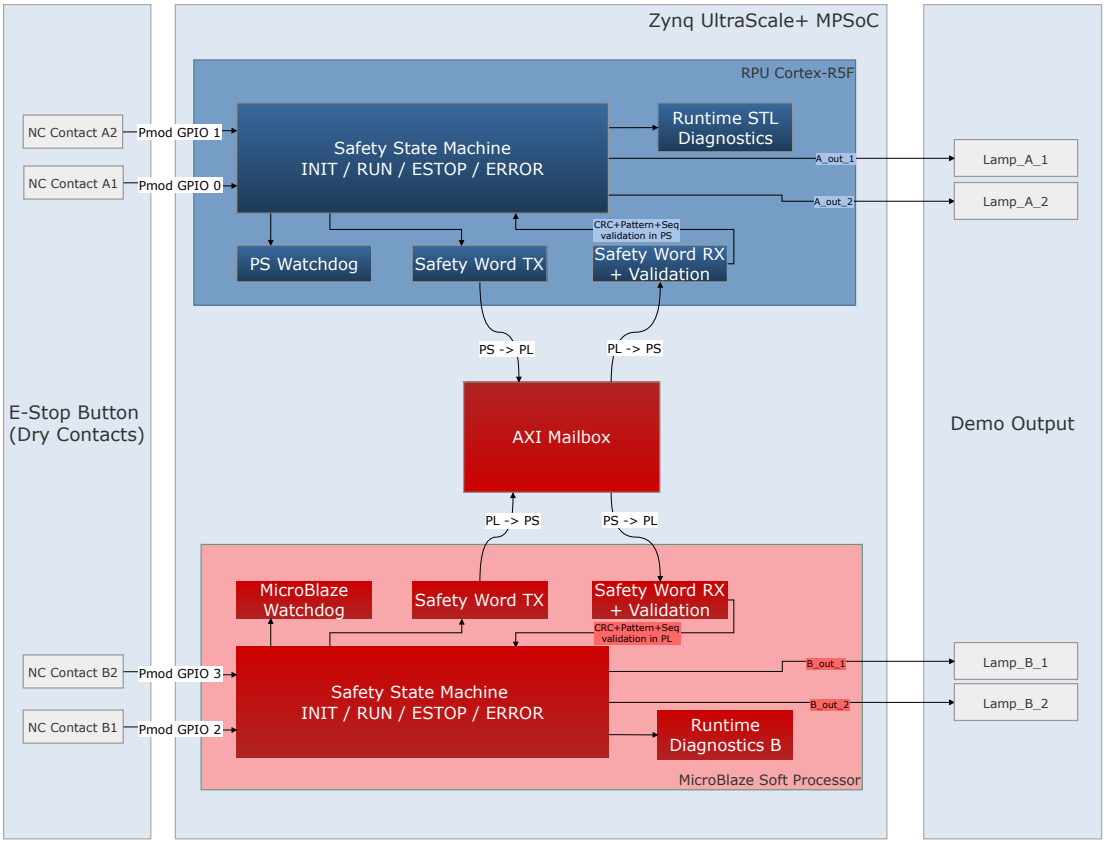

Figure 4.2: Two-channel architecture with independent inputs, local safety logic in each channel, bidirectional safety-word exchange for cross monitoring, and local indicator outputs.

4.3 Channel allocation rationale

To support the intended two-channel safety concept, the safety function must be allocated across two distinct execution environments. This heterogeneous allocation directly follows the safety architecture recommended by the vendor, which suggests combining a processing system domain with a programmable logic domain to fulfill safety requirements on the selected platform.

A highly beneficial consequence of adopting this recommended separation is the inherent architectural diversity it introduces. While the two channels are designed to be functionally equivalent regarding the emergency stop behavior, they are naturally distinct in their realization. By deploying the channels across these fundamentally different execution environments, diversity is organically established in processor architecture, memory organization, timing behavior, and diagnostic implementation. Furthermore, regarding software diversity, the actual source code deployed in each domain is implemented differently to accommodate their distinct hardware abstraction layers and toolchains. This resulting diversity significantly reduces the risk that the same systematic fault or common cause failure will affect both channels simultaneously.

The allocation also reflects the practical conceptual strengths of each domain. A processing system domain inherently provides established platform resources such as advanced timers, interrupts, and watchdog mechanisms, making it a suitable environment for extensive diagnostic tasks. Conversely, a programmable logic domain provides a highly isolated execution context with dedicated logic resources, which is ideal for instantiating an independently managed redundant channel.

4.3.1 Hardware design and specific implementation

Building upon the high-level allocation strategy, Figure 4.3 shows the Vivado block design of the implemented hardware prototype. The targeted hardware platform consists of the processing system and programmable logic. In Vivado, the Zynq UltraScale+ MPSoC IP core is instantiated to implement the Processing System (PS), while a MicroBlaze soft processor is implemented in the programmable logic. This Vivado block design provides the concrete hardware realization of the proposed two-channel prototype.

In the implemented prototype, the PS side, representing Channel A, provides the execution domain together with AXI-connected peripherals used for local interfacing and supervision. As shown in the block design, the PS channel includes AXI GPIO peripherals for E-stop button inputs and LED outputs, an AXI timer for time-based supervision, and a local AXI BRAM controller with associated memory resources. These specific components provide the infrastructure required for Channel A to handle local input signals, maintain runtime behavior, and drive local status indications independently.

Channel B is realized as an independent domain within the programmable logic. The block design illustrates that this channel encompasses the MicroBlaze soft processor alongside its local instruction and data memory, AXI interconnect infrastructure, dedicated AXI GPIO peripherals, a UART interface, and a timer module. By structuring the programmable logic side in this manner, Channel B operates as a self-contained programmable environment that executes its local safety state machine and manages its own input and output functions, fulfilling the role of a diverse redundant channel.

To support the bidirectional cross monitoring requirement established in the architecture concept, a mailbox is integrated into the design. This mailbox serves as the isolated communication path for exchanging safety-related data and status words between the PS side and the programmable logic side. The peripheral contains two 32-bit FIFOs, enabling asynchronous data transmission in both directions. In addition, it routes two separate interrupt interfaces to the respective processors, ensuring that each safety channel is immediately notified when new cross monitoring data is available for validation.

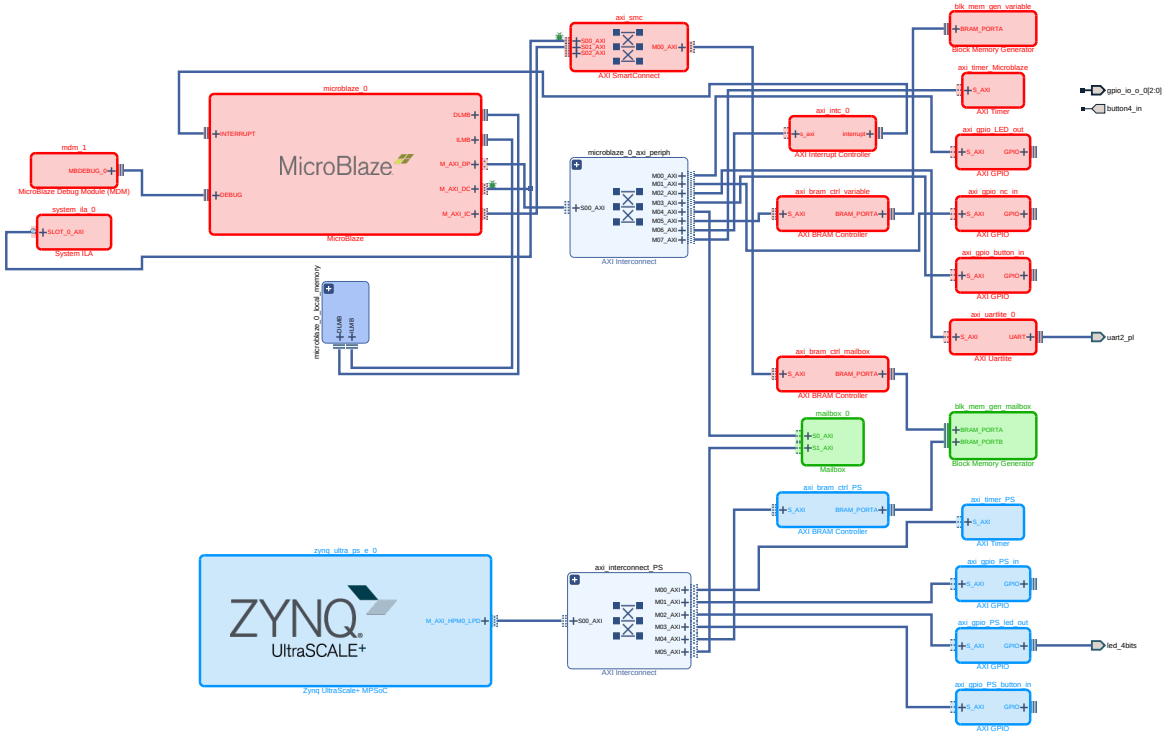

Figure 4.3: Vivado block design illustrating the specific allocation of Channel A in the PS and Channel B in the programmable logic. Blue indicates PS-side resources, red indicates PL-side resources, and green indicates the shared mailbox interface.

4.3.2 Input subsystem

The E-stop is implemented using a latching mushroom-type emergency stop push-button with twist-to-release mechanism, as shown in Figure 4.4. The pushbutton provides two independent normally-closed (NC) contacts, forming a dual-channel input structure. These contacts are routed to both safety channels through separate GPIO access paths, such that each channel can independently sample the E-stop request and interpret its status locally.

A separate manual reset pushbutton is implemented in the same manner and is also connected independently to both channels. In the implemented behavior, reset does not directly resume operation. Instead, it serves as an enabling action for recovery after the E-stop demand has been released or after a faulted condition has been cleared according to the state-machine logic.

Figure 4.4: Mushroom button used for emergency stop actuation [21]

Input acquisition is performed locally in both channels. In both channels, debouncing is implemented in software rather than in external hardware. This supports practical noise filtering and also contributes to implementation diversity between the two channels.

4.3.3 Logic subsystem

Each channel executes local safety logic based on the same functional intent, but with different implementation details. The logic subsystem includes the safety state machine, start-up self-tests, periodic run-time diagnostics, and bidirectional cross monitoring through exchanged safety words.

The implemented state machine is composed of four states: `INIT`, `RUN`, `ESTOP`, and `ERROR`. During operation, each channel evaluates the local E-stop and reset inputs, local diagnostic status, and the received safety status from the peer channel. The two channels are therefore functionally aligned at the safety level, while remaining diverse in processor type, execution model, software structure, and diagnostic realization.

Cross monitoring is realized by periodic exchange of safety words over the mailbox communication path. Each received word is validated using integrity and freshness checks before its payload is used for semantic comparison with the local channel view. A detected communication fault or persistent semantic mismatch is treated as a diagnostic fault and leads to transition into the safe fault-handling path of the state machine.

4.3.4 Output subsystem and redundancy indication

Each channel produces two output indication signals, denoted `A_out` and `B_out`. In the prototype, these signals are connected to indicator outputs to visualize the be-

havior of the redundant two-channel design during E-stop activation, reset handling, and diagnosed faults.

No final hardware-combined safety output is implemented in the current prototype. That is, the channel-local outputs are not combined into a single `OUT_safe` signal through OR logic, voting logic, or another dedicated output-stage mechanism. Additionally, no output readback path is implemented. Consequently, the output subsystem demonstrates local channel reaction and redundancy indication, but it does not yet constitute a complete safety-rated actuation path.

This architectural choice is acceptable for prototype-level validation of the logic and cross monitoring concept, but it also means that the current implementation is stronger as a demonstration of redundant safety decision making than as a final realization of a safety-related output stage.

4.4 Safety state machine

Both channels implement the same four state safety state machine with the states `INIT`, `RUN`, `ESTOP`, and `ERROR`. This model reflects the actual prototype implementation and distinguishes between an emergency stop reaction and a fault condition detected by internal diagnostics. As noted in the architecture description, each channel executes two local state machine instances in parallel, one for each of the two independent NC contacts of the E-stop input. Both instances follow the same state transition rules.

Each state machine instance produces two local output signals, `Out` and `Good`. The `Out` signal represents the local trip request of the channel, while `Good` indicates whether the local logic is considered healthy. The state output mapping of the local safety state machine is summarized in Table 4.1.

Table 4.1: State output mapping of the local safety state machine

State	Out	Good
INIT	0	0
RUN	0	1
ESTOP	1	1
ERROR	0	0

In `INIT`, the channel remains in a start-up condition for a fixed initialization delay of 100 ms. During this phase, initialization and start-up checks are performed before operation is permitted. The outputs remain de-asserted in this state. The 100 ms initialization delay was chosen as a conservative margin to allow start-up, initialization, and cross-monitoring stabilization before normal operation.

After the initialization delay has elapsed, the state machine transitions to `RUN` if the E-stop input is released and no diagnostic fault is present. In `RUN`, the channel is considered healthy and operational, and therefore reports `Out = 0` and `Good = 1`. This is the normal operating state in which the local logic continues to monitor the safety-related inputs, diagnostic status, and cross monitoring information.

If the initialization delay expires while the E-stop input is active, or if an E-stop demand is detected during `RUN`, the state machine transitions to `ESTOP`. In this state, the local trip request is asserted (`Out = 1`) while the channel still reports itself as healthy (`Good = 1`), indicating that the stop request is intentional and not caused by an internal diagnostic failure.

If a diagnostic fault is detected, the state machine transitions to `ERROR`. In this state, the local trip request is de-asserted and the channel declares itself unhealthy, resulting in `Out = 0` and `Good = 0`. Moreover, when a diagnostic fault is detected, the `Good` flag is de-asserted immediately in the same cycle, before the formal state transition to `ERROR` is completed on the following tick. This behavior allows the fault condition to be reflected promptly in the cross monitoring exchange.

Recovery from `ESTOP` does not occur automatically. The E-stop input must first be released, and a separate manual reset rising edge must then be detected before the state machine may return to `INIT`. Recovery from `ERROR` also requires a manual reset, and the system returns to `INIT` rather than directly to `RUN`. This ensures that initialization and start-up checks are repeated before normal operation is resumed. The implemented transition conditions can be summarized in Table 4.2.

Table 4.2: Implemented state transition conditions of the local safety state machine.

Transition	Condition
`INIT` → `RUN`	Initialization delay completed, E-stop released, and no diagnostic fault is present.
`INIT` → `ESTOP`	Initialization delay completed, E-stop active, and no diagnostic fault is present.
`INIT` → `ERROR`	A diagnostic fault is detected.
`RUN` → `ESTOP`	An E-stop request is detected.
`RUN` → `ERROR`	A diagnostic fault is detected.
`ESTOP` → `INIT`	The E-stop is released and a rising edge of the manual reset is detected.
`ESTOP` → `ERROR`	A diagnostic fault is detected while the system is stopped.
`ERROR` → `INIT`	A rising edge of the manual reset is detected and diagnostic fault is cleared.

This state machine structure separates E-stop behavior from internally diagnosed fault handling, while ensuring that both conditions lead to retention of a safe condition until a deliberate recovery sequence has been completed.

4.5 Cross monitoring path

Cross-monitoring is implemented through periodic exchange of safety words between the two channels. Each channel transmits its local safety-related status and receives the corresponding status from the other channel. The purpose of this communication is not to delegate the safety decision to a remote channel, but to support mutual supervision. Each channel continues to execute its own local safety logic independently.

The exchanged safety words provide a means to compare remote and local channel views and to detect corruption, loss of freshness, and semantic inconsistencies. The communication path is implemented through a mailbox-based mechanism. Each channel sends a compact 32-bit safety word at a fixed periodic rate of 5 ms (200 words per second). The 5 ms safety-word transmission period is chosen to provide frequent mutual supervision. On reception, the incoming word is decoded and validated before being used in the cross-monitoring logic.

Because the communication serves a safety-related diagnostic purpose, the design includes integrity and freshness protection within the safety word itself. In addition, each channel evaluates timeouts and semantic mismatches against its own local interpretation.

To prevent false fault latching during asymmetric boot sequences, the cross monitor comparison is gated on start-up completion of both the local and remote channels. Each channel monitors the `SW_FLAG_STARTUP_DONE` flag in the received safety word and also verifies that its own local state machines have left the `INIT` state. Only when both conditions are satisfied does the comparison begin. Until that point, the mismatch window counters are held in reset. This gating ensures that a channel still in its initialisation phase (where `Good` $= 0$) is not incorrectly flagged as faulty by a channel that has already reached `RUN`.

The cross monitoring mechanism therefore contributes both communication integrity diagnostics and mutual supervision between the redundant channels.

4.5.1 Safety word format

A 32-bit safety word is used for the exchanged cross monitoring information. The word is divided into four fields, as shown in figure 4.5

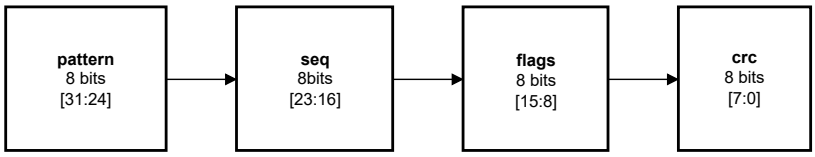

Figure 4.5: Safety word

The pattern value is selected from a fixed 16-entry lookup table, indexed by the lower four bits of the sequence counter. In this way, the pattern changes predictably together with the sequence number and repeats every 16 sequence values. Its purpose is to add a second predictable but changing value beyond the counter itself, thereby improving the ability to detect certain repeated, frozen, or corrupted communication values.

The sequence number is incremented cyclically and provides freshness information. By checking whether the received sequence evolves as expected within an allowed

reception window, each channel can detect repeated frames, missing frames, or unexpected jumps.

The flags field carries application-level safety status. It contains the information required for semantic comparison between channels, such as stop requests, channel health, and diagnostic fault status. This makes the safety word suitable not only for message integrity checking but also for mutual supervision of safety-relevant behavior.

The CRC-8 field is computed over the remaining fields and is verified at the receiver side. This CRC-8 value provides integrity protection for the exchanged safety word. If the received CRC-8 does not match the locally recomputed value, the message is treated as invalid and contributes to a communication fault indication.

Upon reception, each channel performs a complete validation sequence on the received data. This sequence includes CRC-8 verification, pattern verification based on the received sequence number, sequence freshness verification, and a semantic comparison of the received flags with the corresponding local flags. If any of these checks fail, the received data is treated as faulty. Persistent mismatch between the local and remote safety views is handled as a cross monitoring fault and leads to transition into the ERROR state.

4.6 Diagnostic measures

Diagnostics are used to detect abnormal behavior and to trigger safe-state transitions. Diagnostic coverage (DC) is treated as the ratio between the rate of detected dangerous failures and the rate of total dangerous failures. In ISO 13849-1, DC is defined as

$$DC = \frac{\sum \lambda_{D\mathrm{D}}}{\sum \lambda_{D\mathrm{total}}}$$

where

$$\sum \lambda_{D\mathrm{D}}$$

is the sum of failure rates of detected dangerous failures, and

$$\sum \lambda_{D\mathrm{total}}$$

is the sum of failure rates of all dangerous failures. In this work, DC is estimated using the simplified approach in ISO 13849-1 Annex E by selecting appropriate diagnostic measures from Table E.1 for the implemented input, logic, and output diagnostics, and then combining them into an average diagnostic coverage DC_{avg} according to Annex E.2 [4].

4.6.1 Diagnostic requirements for the logic subsystem

ISO 13849-1 Annex E states that if medium or high diagnostic coverage is claimed for the logic subsystem, at least one diagnostic measure shall be applied for each of variable memory, invariable memory, and the processing unit, with each measure achieving a DC of at least 60 % [4]. This requirement directly motivates three complementary diagnostic measures in the proposed design, described below.

Variable memory protection

Each channel maintains a dual-copy representation of all safety-critical runtime variables. The primary copy resides in one physical memory region and the mirror copy in a second, physically separate region. In Channel A, the primary is placed in ATCM and the mirror in BTCM, exploiting the two independent TCM banks of the Cortex-R5. In Channel B, the primary resides in local MicroBlaze BRAM and the mirror in an AXI-attached BRAM. Physical separation ensures that a single-bank failure does not corrupt both copies simultaneously.

At start-up, both memory regions undergo a destructive read or write pattern test using the sequences `0xAAAAAAAA`, `0x55555555`, and `0x00000000`. This test detects stuck-at faults, coupling faults, and addressing faults before any safety-critical data is stored. An address-alias check further verifies that the primary and mirror occupy distinct physical addresses and are not inadvertently mapped to the same hardware resource.

During run-time, a byte-level comparison of the primary and mirror copies is performed on every tick (every 1 ms). Any mismatch is immediately latched as a diagnostic fault and drives the state machine into the ERROR state. This continuous comparison detects corruption caused by wild-pointer writes, stack overflows, or transient memory faults that hardware ECC alone cannot catch. The measure corresponds to the “RAM test” entry in ISO 13849-1 Table E.1 and is applied identically in both channels.

Invariable memory integrity

A CRC-based signature is computed over the invariable memory regions (the `.text` and `.rodata` sections containing executable code and read-only constants) at start-up. This baseline signature is stored and periodically recomputed during run-time. Any deviation between the current signature and the stored baseline indicates that the program image has been altered, for example by a soft error in the memory holding the firmware.

Both channels implement a CRC-64 signature method for memory integrity monitoring. In each channel, the calculated signature is compared with a reference value to detect corruption of the monitored data. This measure is consistent with the signature based diagnostic approach described in ISO 13849-1 Table E.1, including the use of signatures for double words.

Processing unit diagnostics

Both channels implement processing unit diagnostics that correspond to the "self-test by software" measure defined in IEC 61508-7:2010, A.3 and listed in ISO 13849-1 Table E.1. The diagnostic objective is to detect faults in the processor core, its register file, its associated timer and interrupt resources, and the control-flow integrity of the executing software.

In each channel, the processing unit self-test comprises a register consistency check, a timer-counter liveness verification, an interrupt-controller integrity check, and a program-flow monitoring element. The register check compares the current values of safety-relevant processor registers against pre-computed reference values and is executed periodically during run-time. The timer liveness check reads a hardware counter value twice with a known delay and verifies that the counter is advancing, thereby confirming that the timer clock source and counting logic are functional. The interrupt-controller check confirms that the interrupt distributor is enabled and that the timer interrupt line used by the safety application is correctly configured and enabled. Program-flow monitoring is realized through a combination of loop-execution timing checks and temporal and logical monitoring of the logic by the watchdog.

Channel A implements these diagnostics using the Xilinx Safety Test Library (STL) for the Cortex-R5 processor, which provides pre-validated register-check tables for the RPU block, timer-counter liveness primitives, and GIC integrity verification. Channel B implements equivalent diagnostics on the MicroBlaze soft processor through platform specific mechanisms. These mechanisms include a custom register pattern write and read back test that exercises the MicroBlaze general purpose register file. They also include a liveness check based on the AXI Timer and a verification of the interrupt controller status register.

In both channels, any failure of a processing unit self-test is latched as a diagnostic fault and drives the safety state machine into the ERROR state. In Channel A, the STL-based register check executes every 50 ms and the remaining checks are distributed across longer intervals. In Channel B, all three self-test measures execute at a 50 ms interval during run-time.

4.6.2 Watchdog supervision

Each channel is supervised by an independent watchdog mechanism that acts as a liveness guard. The software in each channel periodically services its local watchdog on every main control cycle (every 1 ms). If the software fails to service the watchdog within the configured timeout window, for example due to an infinite loop, deadlock, or processor hang, the watchdog expires and triggers a reset of the affected channel. After reset, the channel enters `INIT` and remains in a safe condition until the defined recovery criteria are met.

The two channels implement watchdog supervision through different mechanisms, reflecting the available hardware resources in each execution domain.

Channel A uses the PS System Watchdog Timer, a hardware countdown timer integrated in the Zynq MPSoC processing system. The timer is configured with a

50 ms timeout window and is serviced at the end of every `ChannelA_Process` cycle. If the timeout expires, the PS watchdog triggers a hardware reset of the Cortex-R5 processor.

Channel B implements a software watchdog within its existing 1 ms timer interrupt service routine. A counter is incremented on each timer tick and is cleared by the main control loop at the beginning of each iteration. If the counter reaches the configured timeout of 50 ms without being cleared, the ISR disables interrupts and executes a branch to the MicroBlaze reset vector, triggering a restart.

The 50 ms interval for watchdog supervision is selected as a prototype-level compromise. It is long enough to limit execution overhead and short enough to supervise execution liveness. The two watchdog instances use different implementation principles and each guards only its own channel. A failure of one watchdog does not affect the other channel's supervision.

In addition to local watchdog protection, each channel indirectly supervises the execution of the other channel through the safety-word timeout mechanism (Section 4.5). If Channel B stops to transmit valid safety words, Channel A detects the absence within 30 ms and transitions to `ERROR`. The 30 ms timeout window is chosen to tolerate transient communication jitter and start-up asymmetry while still detecting faults in a short interval. This cross-channel supervision complements the local watchdog and ensures that a silent failure of either channel is detected by the other within a bounded time.

4.6.3 Input and output diagnostics via cross monitoring

Rather than implementing dedicated input-level or output-level diagnostic hardware (such as test-pulse generators or output readback paths on each GPIO pin), the proposed design addresses input and output diagnostic coverage through cross monitoring of I/O signals and intermediate results within the logic subsystem. This approach corresponds to the measure "cross monitoring of input/output signals and intermediate results within the logic and temporal and logical software monitor of the program flow and detection of static faults and short circuits (for multiple I/O)" in ISO 13849-1 Table E.1.

The mechanism operates as follows. Both channels independently sample the E-Stop input contacts, debounce the signals in software, execute the safety state machine, and produce output decisions.

Each channel encodes its output state and health status into the safety word and transmits it to the other channel via the mailbox. Upon reception, each channel decodes the received safety word, extracts the remote channel states, and compares them point-by-point against its own locally computed states. The comparison covers both the trip request signal (Out) and the channel-health signal (Good) for each of the two internal sub-channels. A persistence filter with a 30 ms window suppresses transient disagreements caused by communication jitter or non-simultaneous sampling. A mismatch that persists for the full window is latched as a diagnostic fault and triggers a transition to `ERROR`.

This cross monitoring arrangement detects several failure modes that affect the input or output paths. For example, a stuck at fault on an E-stop input pin that affects only one channel produces a persistent discrepancy between the local and remote trip states. A corruption of the output decision logic within one channel, such as a register bit flip that changes the computed `Out` value, is detected when the two channels disagree on the trip or `Good` state. In addition, a short circuit or open circuit on a shared input line can produce a mismatch between the debounced input readings of the two channels, which then propagates to a state disagreement detected by the cross monitoring.

Because the cross monitoring operates at the logic level rather than at the physical pin level, it does not replace pin-level diagnostics such as test-pulse readback of individual input contacts or output driver status feedback.

4.6.4 Fault injection interface

A UART command interface is provided on each channel to support on-demand fault injection for demonstration and validation purposes. Commands in Table 4.3 are entered via the serial console and directly set the corresponding diagnostic fault latch, allowing each diagnostic measure to be exercised individually.

Table 4.3: Supported fault injection commands in Channel A and Channel B.

Channel	Command	Injected fault
Channel A	`fi memcrc`	Invariable memory CRC corruption
Channel A	`fi stl`	STL diagnostic fault
Channel A	`fi ram`	Variable memory comparison fault
Channel A	`fi cm`	Cross monitoring mismatch fault
Channel B	`fi memcrc`	Invariable memory CRC corruption
Channel B	`fi ram`	Variable memory comparison fault
Channel B	`fi pu`	Processing unit self test fault

A `fi clear` command clears all locally latched faults and synchronises the dual-copy memory. Each injected fault drives the affected channel's state machine to `ERROR` within one control cycle or within the configured persistence window in the case of cross monitoring faults, providing a deterministic and repeatable means of verifying the end-to-end fault detection and reaction chain.

5

Results

This chapter summarizes the evaluation results of the proposed emergency stop prototype. The presented results cover the implemented safety state machine, the behavior of the redundant two channel architecture, and the effectiveness of the diagnostic and cross monitoring mechanisms. Furthermore, the safety related parameters required for the ISO 13849 standard assessment are calculated. Based on these results, the resulting Performance Level of the prototype is derived and discussed.

5.1 Safety state machine

This section reports the observed behavior of the implemented safety state machine. The prototype was tested under normal operation, emergency stop demand, and diagnostic fault injection to verify state transitions and recovery sequences.

5.1.1 Observed state transitions

Both channels implement the same safety state machine: `INIT`, `RUN`, `ESTOP`, and `ERROR`.

On power up, the system enters `INIT` and remains there for a fixed initialization delay. After the start up checks complete, it transitions to `RUN` and stays there as long as no E-Stop demand or diagnostic fault is present.

When the E-Stop button is pressed, the state machine transitions from `RUN` to `ESTOP` and the safety related output is driven to the safe condition. When a diagnostic fault is injected, the state machine enters `ERROR`. In `ERROR`, the channel declares itself unhealthy. This indicates an internal fault rather than an intentional E-Stop activation.

5.1.2 Safe state retention and recovery

Both `ESTOP` and `ERROR` recover through `INIT` rather than directly to `RUN`, so that the start up checks are repeated before normal operation resumes. The two states differ in their latch conditions.

In `ESTOP`, the trip output remains asserted until the E-Stop demand is released and a manual reset rising edge is detected. In `ERROR`, the channel stays latched until a

manual reset rising edge is detected. On the ERROR-to-INIT transition, all latched diagnostic fault flags are cleared, allowing recovery without a full power cycle.

5.2 Category 3 behavior and redundancy

This section reports results that relate to the intended Category 3 behavior, namely that a single fault in one channel does not lead to loss of the safety function and that some faults are detected by diagnostics.

5.2.1 Redundant fault reaction

In the prototype, a trip request from either channel is sufficient to force the system output to the safe condition. This was observed in two scenarios, when one channel entered ESTOP on an E-Stop demand, and when one channel entered ERROR on a diagnostic fault. In both cases, the system output transitioned to the safe condition regardless of the other channel's state. This behavior supports the Category 3 intent. A single fault in one channel does not prevent enforcement of the safety function, because the remaining channel can still request and enforce the safe state.

5.2.2 Mutual supervision through cross monitoring

Channel B transmits a safety word every 5 ms to Channel A over a mailbox. Each word carries a sequence number, status flags, a pattern, and a CRC. Channel A decodes each message, recomputes the CRC, and checks for mismatch or abnormal sequence behavior.

UART logs captured the raw received safety word, the decoded sequence number and flags, and the fault counter. After the E-Stop was pressed, the outputs of both channels transitioned from 0 to 1 as expected, while the detected mismatch between the two channels remained 0. The sequence number decoded in Channel A continued to increase monotonically without discontinuities, indicating correct periodic transmission and reception of safety words and no detected corruption or loss during the observed test window.

5.2.3 Validation of cross monitoring

Cross monitoring was validated by injecting faults into one channel and observing the reaction of the other channel. Two fault types were used.

The first type was the physical disconnection of a single input. This manual bench action created a persistent discrepancy between the local state of the observing channel and the remote status transmitted by the faulted channel.

The second type was a software-injected diagnostic fault on the remote channel. In these cases, a SafetyRAM compare fault, an STL fault, or a processing-unit self-test fault was injected so that the remote channel declared itself unhealthy by driving Good to 0. The local channel then detected the resulting health-status mismatch through cross monitoring.

Table 5.1 summarizes the injection cases for Channel A. In all four cases, Channel A detected the discrepancy within one monitoring cycle, set its fault flag, and entered `ERROR`. The same tests were then repeated with faults injected on Channel A and observations made on Channel B. The observed results were equivalent.

Across the evaluated cases, the cross monitoring logic detected the intended discrepancies in both directions. Physical disconnection and software-injected health faults produced the same diagnostic reaction. No difference in detection behavior was observed between the two fault types. Within the limits of the injected fault models, these results are consistent with correct bidirectional cross monitoring operation.

Table 5.1: Validation under physical and software.

Injected fault	Method	Result	State	Outcome
Channel B input 1 fault	Physical disconnection of input 1 in Channel B	Channel A detected an inconsistency related to Channel 1, set the fault flag, and drove the corresponding Good signal to 0	`ERROR`	Pass
Channel B input 2 fault	Physical disconnection of input 2 in Channel B	Channel A detected an inconsistency related to Channel 2, set the fault flag, and drove the corresponding Good signal to 0	`ERROR`	Pass
Channel B CH1 health fault	Software injected diagnostic fault on Channel B causing the Channel 1 Good signal to be set to 0	Channel A detected a health status inconsistency on Channel 1, set the fault flag, and kept the corresponding Good signal at 0	`ERROR`	Pass
Channel B CH2 health fault	Software injected diagnostic fault on Channel B causing the Channel 2 Good signal to be set to 0	Channel A detected a health status inconsistency on Channel 2, set the fault flag, and kept the corresponding Good signal at 0	`ERROR`	Pass

5.2.4 Validation of invariable memory test

The invariable memory test was validated by injecting an artificial CRC mismatch at run time via a UART command. When the `fi` command was issued, selected bits of the calculated CRC were deliberately flipped before comparison with the stored reference. The `fi clear` command removed the pending injection.

Table 5.2 summarizes the two test cases. On the first CRC evaluation after injection, the mismatch was detected, the fault flag was set, and the state machine entered `ERROR`. The fault condition was reported through the UART terminal. After the injection was cleared, the system returned to `RUN` on the next reset cycle.

These results confirm that the diagnostic path, which is from CRC computation through comparison to fault latching, reacts as intended to a detected mismatch, without requiring physical corruption of the protected memory.

Table 5.2: Validation of the invariable memory monitoring function under UART-triggered fault injection.

Injected fault	Fault injection method	Observed result	State
CRC mismatch injection	The command `fi` or `fi memcrc` is entered through the UART terminal.	A CRC mismatch is detected, the fault flag is set, and the fault condition is reported through the UART terminal.	`ERROR`
Fault injection clear	The command `fi clear` is entered through the UART terminal.	The UART terminal reports that the fault injection has been cleared.	`RUN`

5.3 Diagnostic coverage estimation

Diagnostic coverage was estimated using the simplified approach in ISO 13849-1 Annex E. Each implemented diagnostic measure was mapped to Table E.1, and the indicative DC values from the standard were used.

5.3.1 Input and output diagnostics

Cross monitoring of input and output signals covers both the input and output subsystems. Per ISO 13849-1 Table E.1, cross comparison of input and output signals with intermediate results in the logic provides an indicative DC of 99% for both subsystems:

$$DC_I = DC_O = 99\% \tag{5.1}$$

Output-stage diagnostic readback via SPI was not implemented in the prototype. The output DC claim therefore rests on the cross monitoring path alone. The implications of this are discussed in Chapter 6.

5.3.2 Logic diagnostics

ISO 13849-1 Annex E requires that when medium or high DC is claimed for the logic subsystem, at least one diagnostic measure must be applied for each of the processing unit, variable memory, and invariable memory. Each measure must provide at least 90% DC.

Processing unit diagnostics combine start-up self-tests, periodic run-time checks, and watchdog-based program execution supervision. Channel A uses the Xilinx STL for the Cortex-R5, which provides pre-validated register checks, TTC liveness, GIC integrity, D-cache scrub, TCM ECC injection. A hardware watchdog (PS System Watchdog) provides independent liveness supervision. Channel B implements a custom PU self-test with register pattern test, timer liveness check, and interrupt-controller verification at 50 ms intervals, together with a software watchdog. The combined PU measures are assessed at 90% for Channel A and 60% for Channel B.

Variable memory is protected by dual-copy RAM with per-tick byte comparison. The primary and mirror copies reside in physically separate memory regions—ATCM

and BTCM on Channel A, local BRAM and AXI BRAM on Channel B. This measure maps to the RAM test entry in Table E.1 and is assessed at 90% for both channels.

Invariable memory is protected by periodic CRC signature verification of `.text` and `.rodata` sections. CRC-64 algorithm is implemented for this verification across both channels. This measure maps to the signature entry in Table E.1 and is assessed at 90% for both channels.

Table 5.3: Logic diagnostic measures and indicative DC per channel.

Measure	Channel A	Channel B
Processing unit (PU)	90%	60%
Variable memory (VM)	90%	90%
Invariable memory (IM)	90%	90%

5.3.3 Average diagnostic coverage

Following ISO 13849-1 Annex E.2, the average diagnostic coverage for the logic subsystem is the arithmetic mean of the three constituent measures:

$$DC_{\mathrm{L}} = \frac{DC_{\mathrm{PU}} + DC_{\mathrm{VM}} + DC_{\mathrm{IM}}}{3} \tag{5.2}$$

The overall average diagnostic coverage combines the input, logic, and output subsystems:

$$DC_{\mathrm{avg}} = \frac{DC_I + DC_{\mathrm{L}} + DC_O}{3} \tag{5.3}$$

For Channel A:

$$DC_{\mathrm{L,A}} = \frac{90\% + 90\% + 90\%}{3} = 90\% \quad \text{(Medium)} \tag{5.4}$$

$$DC_{\mathrm{avg,A}} = \frac{99\% + 90\% + 99\%}{3} = 96\% \quad \text{(Medium)} \tag{5.5}$$

For Channel B:

$$DC_{\mathrm{L,B}} = \frac{60\% + 90\% + 90\%}{3} = 80\% \quad \text{(Low)} \tag{5.6}$$

$$DC_{\mathrm{avg,B}} = \frac{99\% + 80\% + 99\%}{3} \approx 93\% \quad \text{(Medium)} \tag{5.7}$$

Channel A achieves medium DC in both the logic subsystem and the overall average. Channel B's logic subsystem falls in the low range because of the reduced comprehensiveness of its custom PU diagnostics. The overall $\mathrm{DC}_{\mathrm{avg}}$ is nevertheless pulled into the medium range by the 99% input and output contributions. This asymmetry is discussed in Chapter 6.

Although the diagnostic measures are implemented differently in Channel A and Channel B, the overall diagnostic argument is interpreted at the level of the implemented safety function using the input,logic and output decomposition defined in ISO 13849-1. In this view, the input and output subsystems are primarily covered by bidirectional cross monitoring, while the logic subsystem is covered by memory diagnostics, processing-unit diagnostics, and watchdog supervision. The resulting overall assessment is therefore carried forward as a safety-function level DC_{avg} in the medium range.

5.4 Common cause failure (CCF) assessment

Table 5.4 summarizes the common cause failure assessment using the checklist structure of ISO 13849-1 Annex F. For Category 3 architectures, Annex F requires sufficient measures against common cause failures and a minimum score of 65 points.

Table 5.4: CCF assessment according to ISO 13849-1 Annex F.

Measure category	**Applied measure in this work**	**Pts**	**Evidence**
Separation/segregation	The two safety channels are implemented in separated execution environments with controlled communication interfaces and independent local decision paths.	15	Architecture
Diversity	The channels differ in processor architecture, execution model, memory organization, software realization, and diagnostic implementation, reducing susceptibility to common systematic faults.	20	Design rationale
Design/application	Platform-level diagnostic mechanisms include supervision of relevant electrical and thermal operating conditions, and abnormal conditions are treated as diagnostic faults leading to a safe-state reaction.	15	Diagnostics
Assessment/analysis	A structured engineering assessment of potential common-cause failure mechanisms was performed as part of the safety architecture analysis and reflected in the separation, diversity, and diagnostic design choices.	5	Analysis
Environmental	For the intended deployment-oriented architecture, electromagnetic disturbance prevention is addressed through established system-level EMC measures and integration practice. While the present prototype was not subjected to dedicated EMC validation, this aspect is considered in the engineering CCF assessment.	25	Engineering assessment
Total CCF score		**80**	

The estimated CCF score is therefore 80, which exceeds the Annex F minimum requirement of 65 points. The strongest contributors are architectural separation and diversity, which together provide 35 points, followed by protection against relevant electrical and thermal abnormal conditions and the consideration of EMI-related measures in the intended deployment context.

5.5 MTTF assessment

ISO 13849-1 uses $\mathrm{MTTF_D}$ as one input to Performance Level determination. $\mathrm{MTTF_D}$ is related to the failure rate (FIT) by:

$$MTTF_D \text{ (hours)} = \frac{1}{\text{FIT} \times 10^{-9}} \tag{5.8}$$

For continuous annual operation ($8,760$ hours/year):

$$MTTF_D \text{ (years)} = \frac{MTTF_D \text{ (hours)}}{8,760} \tag{5.9}$$

5.5.1 Programmable logic side

Table 5.5 summarizes the programmable logic failure rates by resource type. The total programmable logic failure rate combines permanent and transient contributions:

$$FIT_{\text{total,prog}} = FIT_{\text{perm,prog}} + FIT_{\text{trans,prog}} = 0.194 + 61.07 \approx 61.27 \text{ FIT} \tag{5.10}$$

Substituting into Equation 5.8:

$$MTTF_{D,\text{prog}} = \frac{1}{61.27 \times 10^{-9}} \approx 1.63 \times 10^7 \text{ hours} \approx 1,863 \text{ years} \tag{5.11}$$

5.5.2 PS side

The PS side was analyzed using the FMEDA tool. The permanent failure rate is 2.79×10^{-1} FIT and the transient rate is 6.97×10^{-1} FIT:

$$FIT_{\text{total,PS}} = 0.279 + 0.697 = 0.976 \text{ FIT} \tag{5.12}$$

$$MTTF_{D,\text{PS}} = \frac{1}{0.976 \times 10^{-9} \times 8,760} \approx 116,962 \text{ years} \tag{5.13}$$

5.5.3 Combined system

ISO 13849-1 caps single-channel $\mathrm{MTTF_D}$ at 100 years. Individually, both the PL side (1,863 years) and the PS side (116,962 years) exceed this cap and fall in the *High* range.

The combined system failure rate is:

$$FIT_{\text{total,sys}} = FIT_{\text{total,PL}} + FIT_{\text{total,PS}} = 61.27 + 0.976 \approx 62.24 \text{ FIT} \tag{5.14}$$

This corresponds to a raw $\mathrm{MTTF_D}$ of approximately 1,834 years. After applying the 100-year cap, the system-level $\mathrm{MTTF_D}$ is bounded at 100 years, which falls in the *High* classification.

Within the scope of this assessment, both individual channels and the combined system meet the ISO 13849-1 $\mathrm{MTTF_D}$ criteria for the High range.

Table 5.5: Programmable logic dangerous failure rate summary

Resource type	Total Permanent FIT	Total Transient FIT
Logic	1.594690×10^{-1}	7.430830×10^{-1}
CRAM	3.040500×10^{-2}	3.201872×10^{1}
BRAM	4.185000×10^{-3}	2.831155×10^{1}
URAM	0.000000×10^{0}	0.000000×10^{0}
Total	1.940590×10^{-1}	6.107335×10^{1}

5.6 Performance Level summary

Table 5.6 summarizes the ISO 13849-1 parameters used for the simplified Performance Level assessment of the implemented E-Stop safety function.

Table 5.6: PL parameter summary for the implemented safety function.

Parameter	Result
Category	3
$MTTF_D$ of each channel	High (capped to 100 yr)
DC_{avg} of the safety function	Medium
CCF score	80 (exceeds minimum requirement)
Conservative PL claim	**d**

Figure 5.1 shows the simplified ISO 13849-1 relationship between category, DC_{avg}, $MTTF_D$ of each channel, and the resulting PL. For a Category 3 architecture with medium diagnostic coverage and high $MTTF_D$, the achievable region spans the upper part of PL d and can extend into PL e depending on the exact numerical $MTTF_D$ of each channel.

In the present work, the achieved parameter combination is therefore interpreted conservatively. Although the simplified ISO 13849-1 determination method indicates that Category 3 with medium DC_{avg} and high $MTTF_D$ can reach beyond PL d, this thesis project retains a conservative PL d claim for the prototype. This is consistent with the prototype scope of the work and avoids overstating the result in the absence of a final hardware-combined safety output stage and output readback.

The diagnostic argument follows the input,logic and output decomposition of the implemented safety function. The input and output subsystems are primarily supported by bidirectional cross-monitoring, while the logic subsystem is supported by memory diagnostics, processing-unit self-tests, and watchdog supervision. Although the logic side diagnostic implementation differs between the two redundant channels, the overall assessment remains within the medium DC_{avg} range at the safety-function level. The architectural asymmetry between the channels is discussed in Chapter 6.

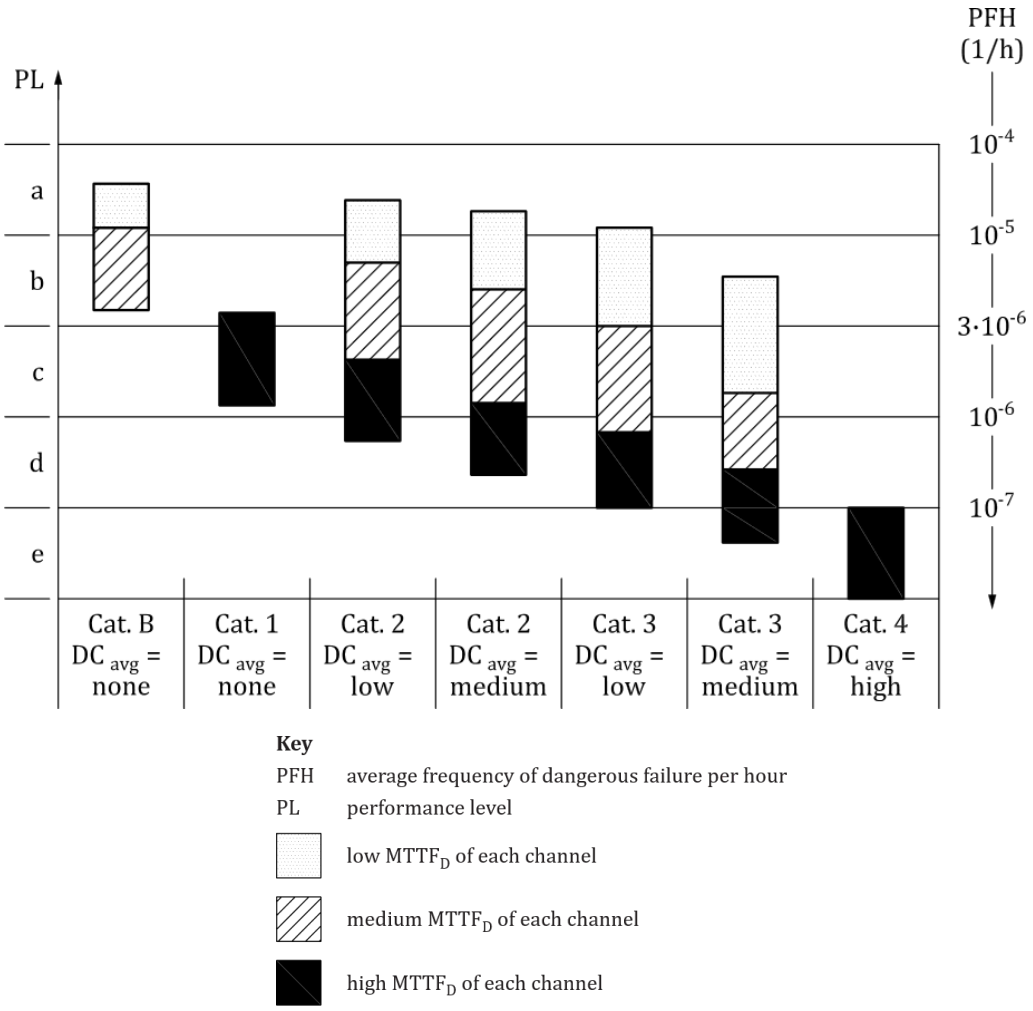

Figure 5.1: Simplified relationship between category, DC_{avg}, $MTTF_D$ of each channel, and PL (adapted from ISO 13849-1:2023, Figure 12) [4].

6

Discussion

This chapter discusses the architectural asymmetries and limitations of the implemented prototype. The validation results that support the observations made here are reported in Chapter 5.

6.1 Architectural asymmetries

The design deliberately introduces diversity between the two channels through processor type, execution domain, and memory architecture. However, several asymmetries go beyond intentional diversity and reflect platform constraints or incomplete implementation.

6.1.1 Watchdog asymmetry

Watchdog asymmetry remains in the current prototype. Channel A is supervised by the PS System Watchdog Timer with a 50 ms timeout and a hardware reset action. Channel B now also operates with a 50 ms timeout, but its watchdog is still implemented in software within the 1 ms timer ISR and performs a soft reset via a branch to `0x0`.

Although both channels now use a 50 ms timeout, the two watchdog implementations are still not fully symmetric in diagnostic capability. Channel B still relies on correct processor execution, correct timer interrupt delivery, and correct operation of the software watchdog logic. An independent hardware watchdog would provide better coverage for certain fault scenarios, particularly faults that prevent interrupt handling or disrupt software execution before the watchdog recovery path can be reached. In addition, the recovery action on Channel B remains a software initiated restart rather than a hardware enforced reset.

In a production implementation, replacing the current mechanism with an independent hardware watchdog, for example an AXI Timer or dedicated supervisory block with a separate reset path, would be necessary to make the watchdog supervision of the two channels more comparable in diagnostic strength.

6.1.2 Processing-unit diagnostic asymmetry

Channel A uses the Xilinx STL for the Cortex-R5, providing pre-validated register checks, TTC liveness, GIC integrity, D-cache scrub, and TCM ECC injection. Chan-

nel B implements a custom PU self-test comprising a register pattern test, timer liveness check, and interrupt-controller verification, executed at 50 ms intervals. While both channels now satisfy the ISO 13849-1 Annex E three-measure requirement for the logic subsystem, the Channel B measures are less comprehensive (no ECC injection, no pre-validated register tables) and have not been independently assessed against a diagnostic test standard.

6.1.3 Diagnostic coverage asymmetry

The two channels reach different logic diagnostic coverage values: 90% (Medium) for Channel A and 80% (Low) for Channel B. The difference originates in the processing-unit measures. Channel A benefits from the pre-validated Xilinx STL, which covers register consistency, TTC and GIC integrity, D-cache scrub,and TCM ECC injection. Channel B's custom PU self-test exercises the register file, timer liveness, and interrupt controller. It is therefore assessed at the lower bound of the acceptable range (60%).

Despite this gap in logic DC, both channels reach the same overall DC_{avg} in the medium range: 96% for Channel A and 93% for Channel B (Section 5.3). The cross monitoring mechanism provides DC of 99% for both the input and output subsystems, and these high values raise the overall average above 90% for both channels. In this sense, cross monitoring compensates for the weaker PU diagnostics on Channel B.

This compensation has a limit. If the PU measure on Channel B were to fall below 60%, the Annex E completeness criterion for the logic subsystem would no longer be satisfied regardless of the input and output DC values. The custom PU self-test therefore carries a disproportionate weight in the overall safety argument for Channel B. Independent validation of that self-test against a recognised diagnostic test standard would strengthen confidence in the assessed 60% value and reduce the risk of an undetected degradation below the threshold.

6.2 Cross monitoring scope and limitations

The cross monitoring mechanism compares logic-level state-machine outputs (Out and Good) between the two channels. It detects corruption of state-machine variables, single-channel input stuck-at faults that produce a persistent trip discrepancy between the two channels, and errors in the output decision logic such as register bit-flips that affect the computed Out or Good value.

The mechanism does not detect faults that affect both channels identically. A common-cause input fault, a shared power-supply disturbance, or a systematic error that corrupts both channels' logic in the same way produces no discrepancy and therefore no cross monitoring reaction. It also does not detect faults downstream of the logic-level decision point, such as a failure in the output driver stage after the Out signal has been computed.

These limitations are inherent in any logic-level cross monitoring approach. The relevant complementary measures include pin-level input test-pulse patterns and

output-stage SPI diagnostic readback. They are described in Chapter 4 and scoped as future work.

6.3 Ethical considerations

This project adhered to standard ethical practices throughout development and reporting. All performance data were obtained from system-internal instrumentation on a bench-level evaluation platform. No personal or user-specific information was collected, stored, or transmitted at any stage of the work.

Generative AI tools were used during the writing phase for language refinement: improving sentence clarity, checking grammar, and reducing repetitive phrasing in the manuscript. These tools were not used to generate analytical content, experimental data, diagnostic coverage assessments, safety conclusions, or any finding that would require engineering judgment. All technical content, including the system architecture, state-machine design, diagnostic measures, fault injection validation, DC estimation, CCF assessment, and MTTF calculation, was developed independently by the authors. Every AI-assisted language suggestion was reviewed and either accepted, revised, or rejected by the authors before including in the final text.

Any implementation details or data subject to confidentiality obligations have either been omitted from this report or presented in anonymised form, in accordance with the relevant policies.

These measures were taken to preserve the integrity and transparency of the work and to ensure compliance with both academic standards and applicable industrial confidentiality requirements.

7

Conclusion

This thesis has presented the design, implementation, and evaluation of a heterogeneous two-channel Emergency Stop prototype on a Xilinx ZCU104 evaluation platform. Channel A runs on the Cortex-R5 in the Processing System and Channel B runs on a MicroBlaze soft processor in the Programmable Logic. Diversity is introduced through processor architecture, compiler toolchain, memory organisation, and timing model.

Bidirectional cross monitoring was realized through a safety-word protocol over a mailbox communication path. Each safety word carries a sequence number, status flags, a pattern, and a CRC-8 integrity field. A persistence filter rejects transient mismatches. Only a discrepancy that persists for the full 30 ms window is latched as a diagnostic fault. A startup gating mechanism prevents false fault latching during asymmetric boot.

Both channels implement the three diagnostic measures required by ISO 13849-1 Annex E for the logic subsystem. Variable memory is protected by dual-copy RAM with per-tick byte comparison in physically separate regions. Invariable memory integrity is verified by periodic CRC signature checks. Processing-unit self-tests cover register consistency, timer liveness, and interrupt-controller integrity. STL is implemented in Channel A, while custom diagnostics are implemented in Channel B. Each channel is supervised by an independent watchdog: a hardware timer on Channel A and a software watchdog on Channel B . A UART command interface enables on-demand injection of all implemented diagnostic fault types, providing repeatable verification of the end-to-end fault detection and reaction chain.

Diagnostic coverage was estimated using the simplified ISO 13849-1 Annex E method. The overall DC_{avg} is 96% for Channel A and 93% for Channel B, both in the medium range. The CCF score of 80 meets the Annex F minimum for Category 3. An $MTTF_D$ analysis of the programmable logic using the XAPP1334 methodology yields a capped value of 100 years (High tier). With Category 3 architecture, high $MTTF_D$, medium DC_{avg}, and a passing CCF score, both channels achieve Performance Level d.

The prototype has several limitations. Channel B uses a software watchdog constrained by blocking UART debug output, and therefore an independent hardware watchdog would be required for a production implementation. Channel B's custom PU self-test is less comprehensive than the STL-based measures on Channel A and has not been independently assessed. No output-stage diagnostics or hardware-combined safe output are implemented. The output DC claim rests on cross moni-

toring at the logic level. Physical output driver faults are not covered by the current prototype. Fault injection has been limited to software-level faults. Physical fault injection and environmental testing have not been performed. Bench testing was conducted at ambient conditions for limited durations; extended endurance testing remains future work.

Future work should address these gaps. Channel B would benefit from an independent AXI Timer-based hardware watchdog with blocking UART output removed from the main loop, enabling a timeout in the 10–50 ms range. The custom PU self-test should be extended with program-flow monitoring and wider opcode coverage, then submitted to independent assessment. The output stage should be completed with smart high-side switches, SPI diagnostic readback, OR-gating for a combined safe output, and an output readback path to close the input-logic-output diagnostic chain. Physical fault injection should complement software-level injection with pin-level stuck-at faults, clock faults, and power disturbances where test equipment permits. The simplified Annex E DC estimation should be replaced with a component-level FMEDA using recognised failure rate databases. Finally, the prototype should be evaluated under temperature, vibration, and EMC conditions with extended run-time testing to characterise rare failure modes.

The implemented prototype demonstrates that a Category 3 style two-channel safety architecture can be realized on a single heterogeneous SoC. Both channels achieve Performance Level d despite asymmetric logic diagnostic coverage: Channel A through stronger processing-unit diagnostics, Channel B through the 99% input and output cross monitoring contributions that raise the overall average. The combination of dual-copy memory protection, CRC-based firmware integrity checks, processing-unit self-tests, independent watchdog supervision, and bidirectional cross monitoring provides a coherent set of diagnostic measures that detect and react to representative faults in the logic subsystem. While the output stage and certain Channel B diagnostics remain at prototype level, the results support the viability of the heterogeneous two-channel approach as a foundation for further development toward a more complete and deployment-oriented safety implementation.

Bibliography

[1] D. Kaur, "The impact of autonomous vehicles on mining operations: Enhancing safety and productivity through technological advancements," *Scholarly Review Journal*, vol. SR Online: Showcase, 10 2024.

[2] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Networks*, vol. 90, 101823, 2019.

[3] C. Buerkle, F. Oboril, M. Paulitsch, P. Popov, and L. Strigini, "Modelling road hazards and the effect on AV safety of hazardous failures," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 1886–1893.

[4] *ISO 13849-1:2023 Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design*, International Organization for Standardization Std., 2023, performance Level (PL) e requirements. [Online]. Available: https://www.iso.org/standard/73481.html

[5] N. Ogorelysheva, A. Vasileva, J. Stadtler, M. Roidl, and F. Howar, "Mitigating emergency stop collisions in AGV fleets in case of control failure," in *2023 Seventh IEEE International Conference on Robotic Computing (IRC)*, 2023, pp. 314–322.

[6] T. Watanabe and M. Itoh, "Validity and considerations of the safety analysis method STAMP/STPA on emergency stop : – case: Unprecedented systems –," in *2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE)*, 2022, pp. 1194–1200.

[7] J. H. Yang, D. J. Kim, W. Y. Choi, and C. C. Chung, "SAFE-STOP system: Tactical intention awareness based emergency collision avoidance for malicious cut-in of surrounding vehicle," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 1533–1540.

[8] Z. Wang, "Research on automatic emergency braking pedestrian system considering road adhesion coefficient," *SAE International Journal of Vehicle Dynamics, Stability, and NVH*, vol. 9, pp. 339–355, 03 2025.

[9] J. Zahálka, J. Tůma, and F. Bradáč, "Determination and improvement of performance level of safety function of emergency stop for machinery," *Procedia Engineering*, vol. 69, pp. 1242–1250, 2014, 24th DAAAM International Symposium on Intelligent Manufacturing and Automation,

2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705814003610

[10] H. Jiang, Y. Wei, W. Li, and J. Li, "Lightweight secure communication scheme based on PUF for in-vehicle controller area networks," in *2024 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2024, pp. 1–8.

[11] G. Capannini, J. Carlson, and R. Mellander, "Reliable visibility algorithms for emergency stop systems in smart industries," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2022, pp. 89–94.

[12] M. Wu, F. Guo, J. Wu, Y. Xiao, M. Jin, and Q. Zhang, "Emergency stop system of computer vision workstation based on GMM-HMM and LSTM," in *2023 9th International Conference on Automation, Robotics and Applications (ICARA)*, 2023, pp. 150–154.

[13] F. Duerr, J. Ziehn, R. Kohlhaas, M. Roschani, M. Ruf, and J. Beyerer, "Real-time global optimization of a fail-safe emergency stop maneuver for arbitrary electrical/ electronical failures in automated driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–8.

[14] A. Porras-Vázquez and J.-A. Romero-Pérez, "A new methodology for facilitating the design of safety-related parts of control systems in machines according to ISO 13849:2006 standard," *Reliability Engineering & System Safety*, vol. 174, pp. 60–70, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832017308359

[15] G. Kaczor, M. Szkoda, and M. Machno, "Hazard and risk analysis of railway vehicle control systems according to safety integrity levels," *Transport Problems*, vol. 18, p. 179, 04 2023.

[16] A. Allouch, A. Koubâa, M. Khalgui, and T. Abbes, "Qualitative and quantitative risk analysis and safety assessment of unmanned aerial vehicles missions over the internet," *IEEE Access*, vol. 7, pp. 53 392–53 410, 2019.

[17] H. Bao, K. Wang, X. Wei, Y. Cheng, M. Zhao, and G. Li, "Highly redundant safety scheme and its experimental validation for the electronic-controlled pneumatic braking system of commercial vehicle for autonomous driving," *IEEE Access*, vol. 12, pp. 138 201–138 210, 2024.

[18] L. Veneroso, A. Anticaglia, L. Lelli, A. Mosci, C. Annicchiarico, and R. Capitani, "Open ECU EPS system, preliminary design using a hardware-in-the-loop approach," *IOP Conference Series: Materials Science and Engineering*, vol. 1306, no. 1, p. 012028, may 2024. [Online]. Available: https://doi.org/10.1088/1757-899X/1306/1/012028

[19] K.-S. We, C.-G. Lee, J. Lee, K. Kim, K. Yi, and J.-C. Kim, "ECU-in-the-loop real-time simulation technique for developing integrated vehicle safety system," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, 2014, pp. 1432–1437.

[20] Xilinx, Inc., *ZCU104 Evaluation Board User Guide (UG1267)*, Xilinx, Inc., October 2018, v1.1. [Online]. Available: https://docs.amd.com/v/u/en-US/ug1267-zcu104-eval-bd

[21] TE Connectivity, *Series ES Emergency Stop Switches*, 2023, data sheet. [Online]. Available: https://www.te.com/content/dam/te-com/documents/industrial-and-commercial-transportation/global/kissling-documents-data-sheets/Data_Sheets_Documents_Kissling/switches/ict-kissling-SW-ES-K1166720-ds-eng-2023-estop.pdf