Validation of Functional Safety Module with IEC 61508 Safety Standard

Major Project Report

Submitted in fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(Embedded Systems)

by

Aayushi Kothari

(18MECE01)

Electronics & Communication Engineering Department

Institute of Technology

Nirma University

Ahmedabad-382 481

Validation of Functional Safety Module with IEC 61508 Safety Standard

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

by

Aayushi Kothari
(18MECE01)

Under the guidance of

External Project Guide:

Mr. Mahaveer Addanki

System Validation Engineer

Intel Technology India Pvt. Ltd.,

Bengaluru.

Internal Project Guide:

Dr. Ruchi Gajjar

Assistant Professor EC Department,

Institute of Technology,

Nirma University, Ahmedabad.

Electronics & Communication Engineering Department

Institute of Technology-Nirma University

Ahmedabad-382 481

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in Embedded Systems at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

Aayushi Kothari

18MECE01

Disclaimer

"The content of this thesis does not represent the technology,opinions,beliefs, or positions of Intel Technology India Pvt. Ltd., its employees,vendors, customers, or associates."

Certificate

This is to certify that the Major Project entitled **"Validation of Functional Safety Module with IEC 61508 Safety Standard"** submitted by **Aayushi Kothari (18MECE01)**, towards the partial fulfilment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for the award of any degree or diploma.

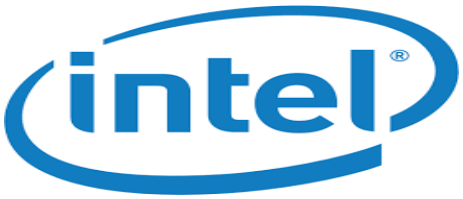
Date: **Place: Ahmedabad**

Dr. Ruchi Gajjar
Internal Guide
Nirma University

Dr. N.P.Gajjar
Program Coordinator
Nirma University

Dr. Dhaval Pujara
Head of Department,
Nirma University

Director
Institute of Technology
Nirma University



Certificate

This is to certify that the Major Project entitled **"Validation of Functional Safety Module with IEC 61508 Safety Standard"** submitted by **Aayushi Kothari (18MECE01)**, towards the partial fulfilment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

Ms. Yamuna Lingam
Engineering Manager
Intel Technology India Pvt. Ltd.
Bangalore

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr. N.P.Gajjar**, PG Coordinator of M.Tech Embedded Systems and **Dr. Ruchi Gajjar** for guidelines during the review process.

I take this opportunity to express my profound gratitude and deep regards to **Dr. Ruchi Gajjar**, guide of my internship project for her exemplary guidance, monitoring and constant encouragement.

I would also like to thank **Mr. Mahaveer Addanki and Ms. Yamuna Lingam**, external guide of my internship project from **Intel Technology India Pvt. Ltd.**, for guidance, monitoring and encouragement regarding the project.

Aayushi Kothari

18MECE01

Company Profile

Intel Corporation was founded on 18 July 1968. It is an American multinational corporation and technology company which has its headquarter in Santa Clara, California, in Silicon Valley. It is the world's second-largest and second highest valued semiconductor chip manufacturer based on revenue and is the inventor of the x86 series of microprocessors, the processors found in most personal computers (PCs).

Our in-depth understanding of technology and hands-on experience helps us to bring the best technical practices and to deliver a world-class quality of services. Continuous learning and an innovative attitude are part of our work culture that always keep us prepared for best effective practices.

We serve our clients with silicons(CPU), firmware and different technologies developed by Intel loaded to it along with its validation done.

Contents

List of Figures

List of Tables

List of Abbreviations

FuSA	Functional Safety
IOT	Internet of Things
FW	Firmware
SW	Software
IFWI	Integrated Firmware Image
BIOS	Basic Input/Output System
SDLC	Software Development Life Cycle
STLC	Software Testing Life Cycle
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ASIL	Automotive Safety Integrity Level
SIL	Safety Integrity Level
EUC	Equipment Under Control)
UAT	User Acceptance Testing
LLD	Lower Level Design

SOC	System on Chip
FPGA	Field-Programmable Gate Array
SLC	Safety Life Cycle
1oo1	One out of one
1oo2	One out of two
1oo2D	One out of two with Diagnostic
SIS	Safety Integrated System

Abstract

In the current generation, humans are using a lot of machines and embedded devices which can cause harm with either physical loss or financial loss. And these machines add luxury to our lifestyle, so it became a crucial part of our life. It's necessary to make our system hazard-free or at-least to diminish the risks which are introduced due to it. So, a product must be developed considering the risk which is needed to be reduced to its tolerable value. This thesis gives vision to the plan and develops the safety-critical systems which implement the safety functions to decreases the risks caused by different machines and systems. The probability of occurrence or severity of incidents is deteriorated by tracking the system in such a way which keeps the failure in control. The regulation of law and Standard plays a vital role in designing and developing such a system which follows all the requirements starting from initial phase to final product level. In this thesis, Functional safety system is introduced with a design viewpoint for the development of a product which will be used in the industry. A Software-based Lockstep comes in a fault-tolerant system that performs the same set of operations at the same time in a parallel fashion. The comparator logic compares the result on a cycle-by-cycle and if the outcomes are equivalent. On the off chance that there are disparities between the outcomes, this could be a captured without causing failure to the whole system. To increase the correctness of Software-based Lockstep, required set of scenarios are set to validate the feature using multi-threading, Inter-Process Communication and Scheduling processes. Thus, Validation of such timely critical Software-based Lockstep will help to detect the failure in Functional Safety Module of Intel SOC.

Chapter 1

Introduction

1.1 Motivation

Functional safety is a concept which is applicable across all industry sector to enable the multiform technology that aims Safety-related systems. The leading goal defining Functional Safe System is to turn down the probability of collapses or failures at a given tolerable rate in the spectre of malfunctioning activity. Safety Standards provide a reference life-cycle to achieve Functional safety of E/E/PE i.e. Electronic/Electrical/Programmable Electronic systems based on Hazard identification and risk analysis.

1.2 Objective

Functional Safety is becoming a crucial issue for Industrial automotive Domain which have its Standards IEC61508 and ISO26262 respectively. Any Hazard caused by malfunctioning of Electrical and electronic or programmable Devices due to Random Hardware Failure and Systematic Failure is addressed by these Standards. IEC61508 plans the overall requirements to confirm the system's operation, implementation, planning, designing, maintenance is delivering the needed Safety Integrity Level (SIL). There are 4 SILs which states according to the risks convoluted

in the application, and to shield in case of the highest risks, SIL4 being used.

1.3 Problem Statement

The system is being developed based on the Functional Safety Standards of IEC 61508 to make sure that the product meets out its functional safety requirements. Verification and Validation model techniques being used with each of the design phases to secure that the system is developed as per the given safety requirements. The End product makes sure that the system is safer to use without causing any hazards due to malfunctioning behaviour of electrical or electronic devices because of Systematic and Hardware failure.

1.4 Requirements

For the implementations following the knowledge requirement is needed:

- Functional safety Standards.
- Knowledge of Embedded C language.
- Linux Scripting
- Knowledge of Multi-processes and Multi-threading.
- Mailbox concept

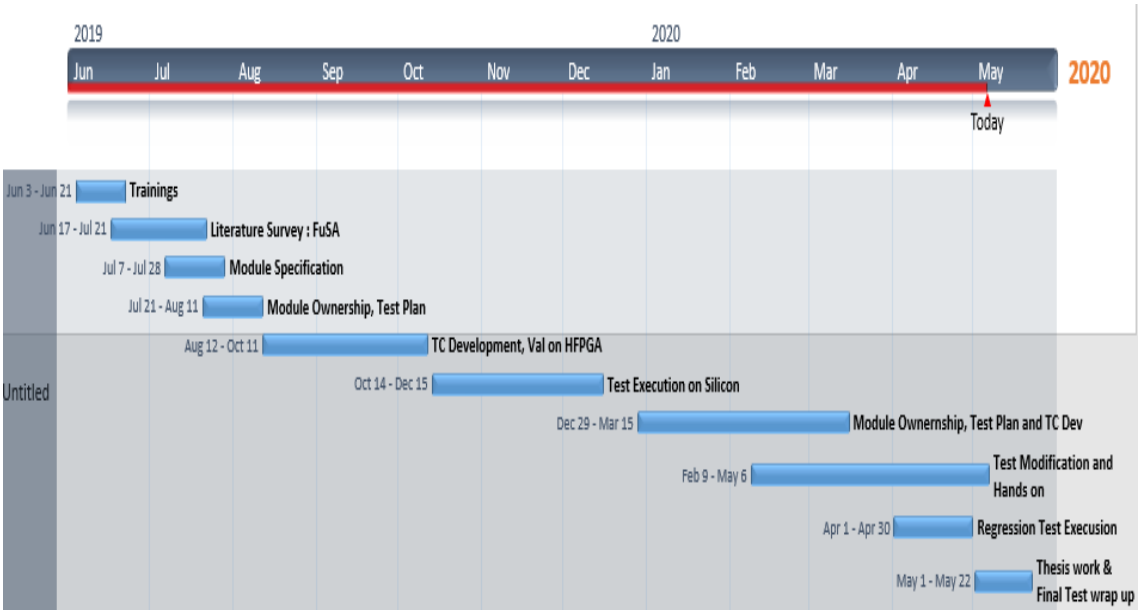
1.5 Scope of Work

The focal scope of work is to reduce the observable risk associated with functional failure of the system below a threshold given by the assessment of severity, exposure, and controllability with efficient and optimised Validation and Verification Techniques.

1.6 Gantt Chart

1.6.1 Internship Summary

This Gantt Chart shows the work done throughout the Internship.



1.7 Thesis Organization

- **Chapter-1 Introduction** points the quick requirements, Problem statement also give a brief idea about project goal and the origin of motivation. As part of the work done is manifested with the Gantt charts pointing project development workflow done throughout the internship.

- **Chapter-2 Background Theory** points the Background Theory of Functional safety and its Standard and its importance in Safety systems.

- **Chapter-3 Architecture of Functional Safety Module** describes the overall system architecture in brief. In this chapter, a description of Intel architecture functional flow is mentioned along with an upper-level flow of

Boot process of a system and how the next processes are carried out has been explained. Also explained the different fault mechanism detection methods and how the software-based Lock-stepping is used in industrial application. At last, the IFWI Stitching of particular feature is also mentioned.

- **Chapter-4 Software Validation Techniques** describes the different techniques and modelling in development and Validation environment which is being practised in the real world. Also discussed the pros and cons of the modelling Technique used. Also, the work was done as part of Validation.

- **Chapter-5 Results and Outcomes** describes about the Validation cycle and Test case development also about the coverage details. All together discuss about the all the modules which are being owned in this project.

- **Chapter-6 Use Case** describes the general and how the Internet of Things works in the industry.

- **Chapter-7 Conclusion** describes the executive summary of this thesis and expected future work is pointed out in this chapter.

Chapter 2

Background Theory

This section includes a history of safety, systems which are related to safety and their engineering. Also described the Functional Safety significance to the extent of the design of a system.

2.1 Background

In public spaces, Industries, corporate; all are encircled by a growing extend of electric and electronic devices and systems. Perhaps in some situations even knowing about the unwanted accidents some possibility of loss remains irrespective of actions undertaken to lessen the risk occurrence. For instance, present-time cars have many techniques to reduce the risk ratio of injury or loss of life in any of the unwanted accidents but some of the methods cannot make the system completely failure-free or totally safe system. So, in order to make the safe system, only thing is proposed is to reduce the risk or failure scenario into a tolerable level which nor give worse impact to life or surrounding and suffice the risk. From an ideal safety point, there should be a conformation which indicates the risk is been mitigated to a tolerable level.

2.2 Terms and Definitions on Safety Concepts

In this section, basic terms and definitions are described.

2.2.1 Fault

When a failure occurs, the item enters the failed state. A failure may occur in two phases of stages i.e. while running or while in Standby.

Differences between a computed, observed, or calculated value or strategy and the acceptable, specified, or theoretically correct data or condition. It occurs when the performance of a system deviates from the target.

2.2.2 Failure

Failure is a situation that occurs at a particular point in time. It may develop gradually and it occurs as a sudden unwanted event. There are different possible ways of failure getting disclose which may be on-demand, or during a functional test or by monitoring or Diagnostics[3].

2.2.3 Relationship between Failure, Fault and Error

A failure initiates from an error. When the failure happen, the item comes under a fault state.

2.3 Failure category

Failures may be classified according to their:

a. **Causes:** To avoid future occurrences and make judgments about the repair.

b. **Random failure**: The failure, observed at a random point of time, which results from multiple attainable degradation logics in hardware. Random hardware failures may be characterized by a failure rate that is either:

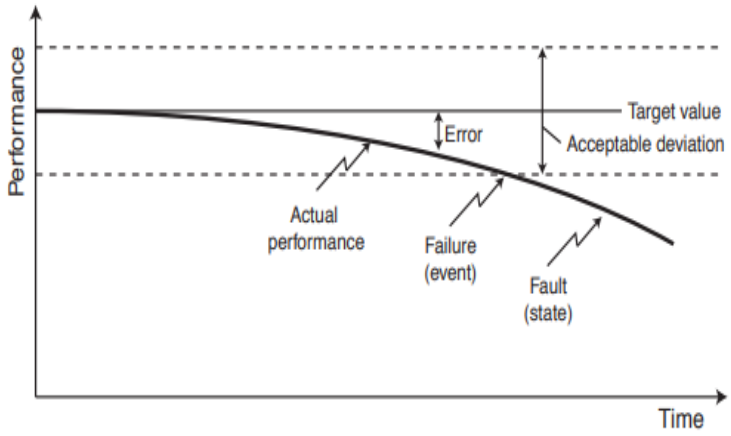


Figure 2.1: Relationship between Error, Fault and Failure

(1) Constant, meaning that the component is in its useful life where impact of ageing is negligible.

(2) Non-constant, meaning that the component is subject in the burn-in a phase of a wear-out phase

- **Systematic Failure** : Failure which are certain cause, that can be removed by any altercation of the System Design and in the change in the manufacturing methods, change in operational process, documentation or any related factors.

  (1) It is always repeated when triggering condition is available.

  (2) Systematic faults may be introduced in any life-cycle phase.

  (3) If properly corrected, the failure will in theory never re-appear

c. **Effects :**To rank between critical and not so critical failures.

- **Safe failures**:

  (1) Results in the pretended operation of the safety function to keep the EUC in a safe state or maintain a safe state condition.

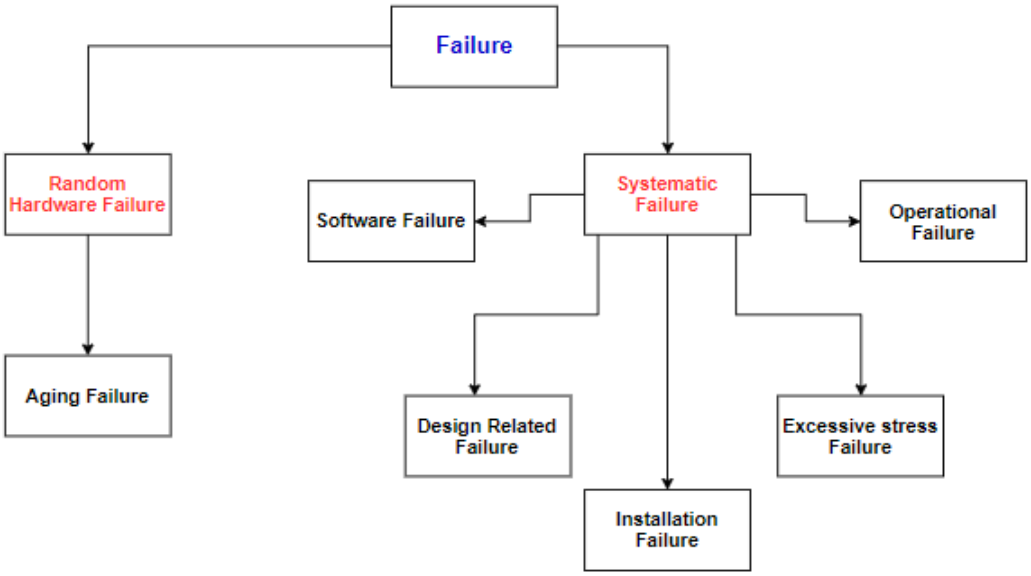


Figure 2.2: Failure Classification

(2) Also, gain the possibility of the mock operation of the safety function to keep the EUC into a safe state or maintain a safe state. A safe failure can end up in a drop in production or service.

- **Dangerous failures** :

(1) Avoid a safety action from operating when required or causes a safety function to fail such that the EUC is kept into a hazardous or potentially hazardous state.

(2) Reduces the possibility that the safety function works correctly when required

(3) A dangerous failure may result end up with a loss of safety.

d. **Detectability:**To distinguish failures that may be revealed "automatically" and those that may be hidden until the special effort is taken, such as proof tests.

- **Detected**: Failure which is immediately evident to operation and main-

tenance personnel as soon as it occurs. A typical example is failures reported as diagnostic faults or alarms.

- **Undetected** : Failure which is not immediately observed in operations and in any maintenance cycle. A typical example is a failure that is hidden until the component is asked to carry out its function.

2.4 Functional Safety

Functional safety explains "absence of unreasonable risk due to hazards caused by malfunctioning behaviour of EE systems"[4]. It means, that a failure in one part of the EE may lead to accidents with damaged equipment, injuries or even the loss of lives. Functional safety is not to confuse with active safety or passive safety. Active safety is grouping features and technology with a focus to avoid accidents. Passive safety covers subsystems that aim to reduce the impact of an accident. Both active and passive systems must be functionally safe as well, since a malfunctioning behaviour may result in accidents[2].

The goal of the functional safety standards is to provide methods to recognize and avoid random failures for electronic Devices and systematic failures for software. The standards propose a development process requires, for example, specific methods to be applied for development, independent verification and validation and collection of safety arguments. The whole SLC i.e. Safety Life Cycle is described by IEC 61508. This SLC assumes that products are developed from scratch and a V-Model-based development process is used. When developing safety-critical products in product lines, this SLC is valid and compliance needs to be shown when certifying the products.

2.4.1 Functional Safety Standard: IEC61508

The concepts of functional safety and SIL i.e. Safety Integrity Level of electronic programmable devices and electrical are fulfilled in the issued Functional safety Standard of International Electrotechnical Commission IEC 61508, Functional Safety of Electrical/Electronic/Programmable Devices which gives the in-depth guide to how to target functional safety from the very beginning of product to its system-level and also provides the requirements with full life-cycle of system.

The main objective of the standards is the theory of safety life cycle, risk and safety logic, safety integrity levels. The safety life cycle is described as an Engineering system which involves necessary processes to attain functional safety. A function of the frequency of the hazardous event and the event consequence severity is the risk. Safety integrity levels indicate the target Value of safety functions to be executed by E/E/PE safety-related system [1].

This IEC61508 functional standard which has SILs are explained with respect to the risks implicated in the customised application, where SIL4 is used to give protection against the highest risks. This Standard is communicated with all the links with a specified process which have common terminology and parameters to look into.

The Standard is divided into 8 parts:

- IEC 61508 - Part 0, for Functional safety and IEC 61508
- IEC 61508 - Part 1, for General requirements
- IEC 61508 - Part 2, for Requirement for E/E/PE safety-related system
- IEC 61508 - Part 3, for Software-based Requirements
- IEC 61508 - Part 4, for Definitions and its Abbreviations

- IEC 61508 - Part 5, for Examples and Techniques for the determination of safety integrity levels
- IEC 61508 - Part 6, for Guidelines on the use of IEC 61508-2 and IEC 61508-3
- IEC 61508 - Part 7, for Overview of techniques and methods

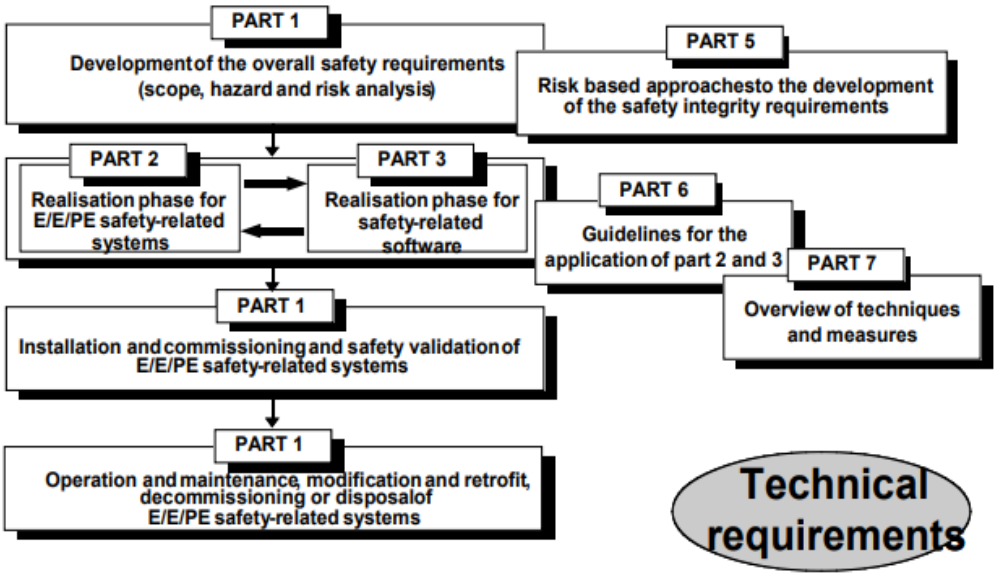


Figure 2.3: Technical Requirements of IEC 61508 [1]

Part 1 covers the overall requirements for documentation, management, compliance of functional safety and its assessment. It gives a detailed presentation of the life-cycle of safety Standard.

Part 2 includes the basic Hardware-based requirements of safety-related systems. this part covers the entire system development with respect to the development of hardware product. Also gives the different techniques to handle failures during run-time operation. It incorporates the hardware-based fault tolerance, diagnostics requirements, limitations, systematic safety issue for hardware.

Part 3 indicated the Software related requirement for IEC 61508 standard. It comprises all software details of the safety life cycle. It contains the list of techniques and measurement used in software development.

Part 4 comprises the abbreviations and definitions been used in the whole standard. it is used for the reference to get the accurate meaning of the terms used in Standard.

Part 5 includes the Details for the risk, safety integrity, SIL selection and tolerable risk. the quantitative and qualitative methods are used for SIL selection. the quantitative approach included the frequency of occurring hazardous incident with respective of the strength of the consequences for any situation.

Part 6 covers the guideline of Software and Hardware requirements including the in-depth information of the procedures. It provides the method of calculations for estimating the multiple failure modes within architecture.

Part 7 covers methods and techniques to maintain safety even in error injecting conditions. It includes the prevention of system failure or any failure component throughout the safety life cycle.

This Standard is field application and hs been used a for industries measurements, control and automation. It gives comprehensive specifications for both Software and Hardware with enclosed hardware and system integrity. An emergency shutdown plays a vital role in ensuring the safety of operation and production which is given by Safety Interlock System SIS. Similarly, the ISO 26262 Standard is for Road vehicle and automation which is the modification of IEC61508. There is also sub-partition in IEC61508 standard where a particular standard is used for a specific sector or area.

- IEC 61511 in Process Industry
- IEC 61513 in Nuclear power plant
- IEC 62061 in Machinery Section
- IEC 61800-5-2 in Power drive systems

2.4.2 Risks and Risk Reduction

IEC61508 has some of the following observation on risks-

a. Risk cannot be reduced to Zero or no risk.

b. Risk which is not tolerable should be brought down to tolerable or below that.

c. Safety should be considered from the initial phase.

2.4.3 Hazard and Risk Analysis

The IEC61508 standard conveys that "**The EUC (Equipment Under Control) risk shall be eliminated or evaluated for each determined Hazardous event.**"

The analysis of Hazard is carried by six different categories of occurrence and four different consequences is shown in the below tables.

Table 2.1: Category of Frequency[1].

Category	**Definition**	**Failures per year**
Frequent	Many times in system lifetime	$10^{-}3$
Probable	Several times in system lifetime	$10^{-}3 - 10^{-}4$
Occasional	Once in system lifetime	$10^{-}4 - 10^{-}5$
Remote	Unlikely in system lifetime	$10^{-}5 - 10^{-}6$
Improbable	Very unlikely to occur	$10^{-}6 - 10^{-}7$
Incredible	Cannot believe that it could occur	$10^{-}7$

Table 2.2: Category of Consequences[1].

Category	**Definition**
Catastrophic	Multiple Loss of life
Critical	Loss of single Life
Marginal	Major injuries to one two persons
Negligible	Minor injuries at worst

2.4.4 Safety Integrity Level - SIL

SIL is the possibility of safety based system executing the necessary safety actions within allocated time duration in all conditions. It is the discrete level for the specification of the safety integrity requirements. On the basis of the number quantitative factor and qualitative methods such as development process and safety life cycle management, the SIL is determined.

Following table describes the SIL levels and its respective failure probability.

Table 2.3: Classification of SIL[1].

SILs	**Low demand mode of operation**
4	$10^{-}5 - 10^{-}4$
3	$10^{-}4 - 10^{-}3$
2	$10^{-}3 - 10^{-}2$
1	$10^{-}2 - 10^{-}3$

So, SIL4 is used to give protection against the highest risks and SIL1 is used to give protection against the lowest risks.

2.4.5 Functional Safety Life-cycle

The IEC60518 standard life-cycle describes the entire production Steps. The process begins with the basic objective, concept, the scope of the system along with system assessment. Further risks are examined the risk mitigation work. The Functional safety system is selected to reduce such situation.

Functional Safety System process initiates with proper Scope Definition and its conceptual requirement specification to define the project. Then, according to the safety requirement plotted, hardware and software design is developed and finally, both domains are integrated into a functional unit and validated as a whole functional system. On the basis of the failure and risk analyses, the modifications are introduced from where the whole process repeats to make accurate Functional safety system.

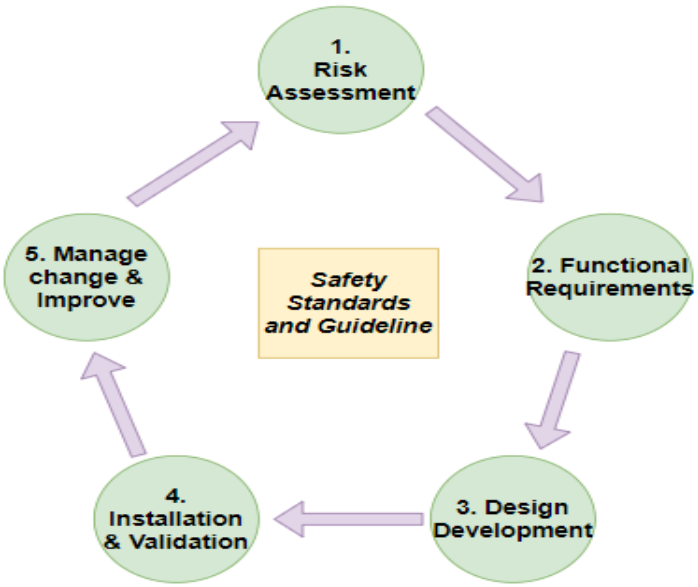


Figure 2.4: Life cycle of IEC61508

2.4.6 Fault Avoidance and Fault Tolerance

Functional safety Standard ensures a process that identifies the possible faults to eliminate it and identify it during Run time. It can be avoided by designing the system in such a way that it does not even occur. But the fact is all faults are not eliminated but it can be detected during a run time which keeps the systems in a predefined safe state.

The following Figure explains the basic flow of Fault detection and outstretches to the safe-state as explained in IEC61508 Standard.

Initially System running in Normal operation and Fault turned up where a fault may be noticed in a diagnostic interval using Diagnostic logic and puts the system in Safe State during Fault Reaction time. A fault-tolerant system is a system where the system is kept in a safe State even before the occurrence of Hazard which causes harm. It is necessary to get a clear flow to avoid faults and also able to detect the fault in run-time and reach it to a safe state. The technical concepts should mention the need for analysis of fault, its avoidance, residual faults which are detected in a

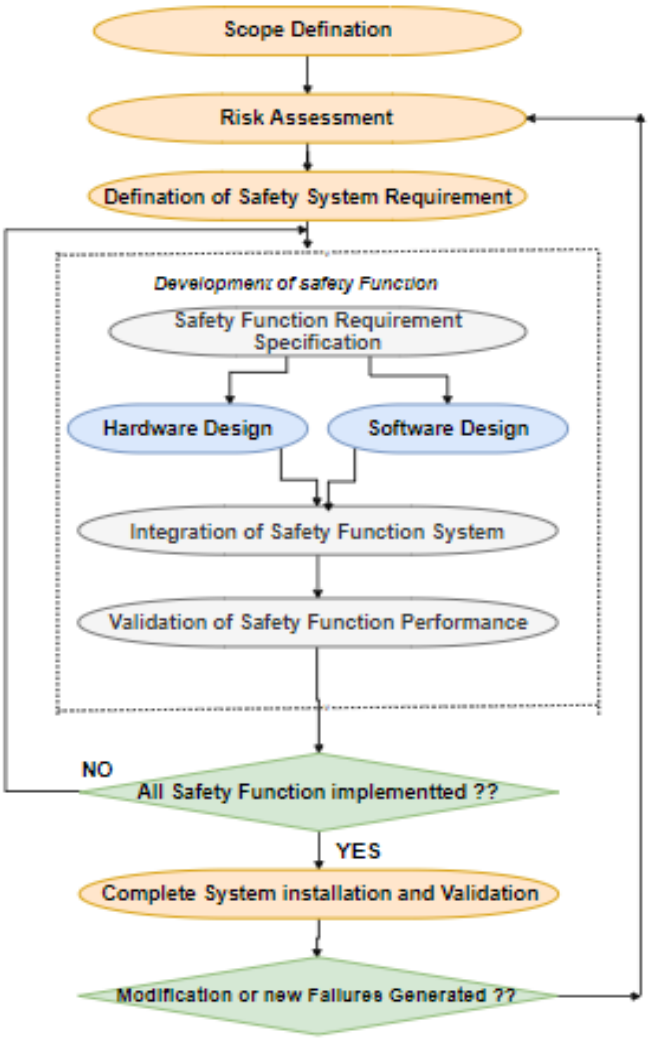


Figure 2.5: Simplified processes for Safety System stated by IEC61508

safe state for all configuration of all products.

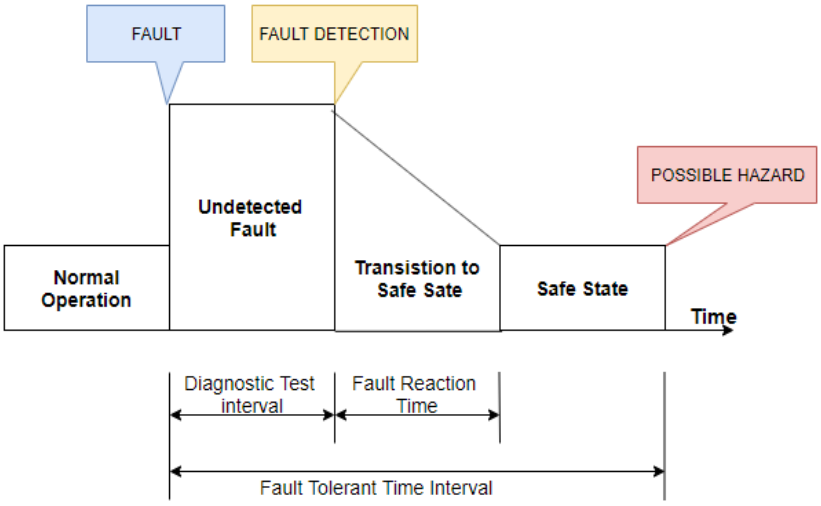


Figure 2.6: Fault Tolerance Flow of IEC61508

Chapter 3

Architecture of Functional Safety Module

This chapter describes the Functionality of Firmware and Software level flows at the system level. This defines the design and architecture of module meeting Functional Safety Requirements and its functionality.

3.1 Hardware to Host Application Architecture

in any Embedded system, BIOS is required that is Basic Input Output System where the heart of BIOS is Boot ROM. It will bootstrap the processor and batch the OS which will enable the fundamental ports at Boot time. the maintenance of the HW activity is done by it until the OS encounter to take the control. A BIOS is in-charge for initialling those memory registers to reach into the discovery phase. After this step, the whole control is handed to OS. Its aimed to make the system up and keep it running where it also look after power unit management. On the motherboard, the Bios is saved as Firmware Image. Its Function is to initialise memory register, processor, other peripherals and chipset.

The register setting is captured in a non-volatile chipset where the User can modify some of the settings by BIOS console.

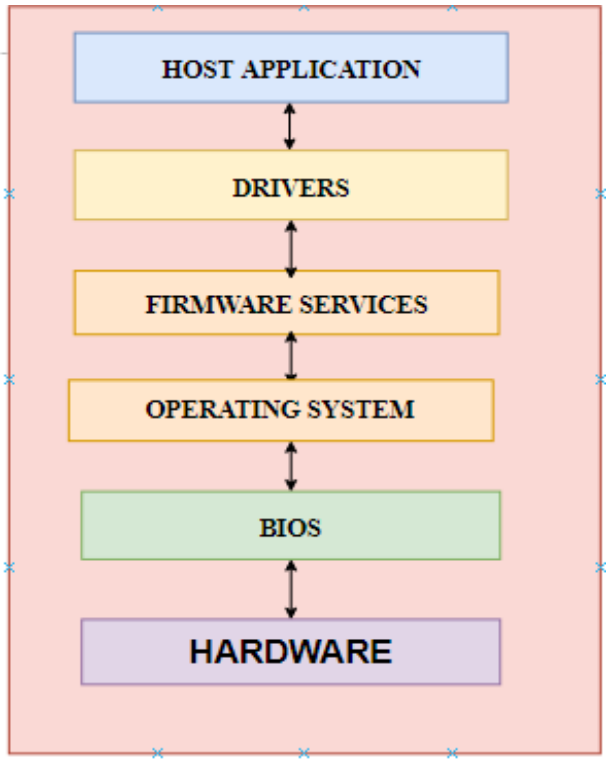


Figure 3.1: Layered Architectural Flow

BIOS is stored as Firmware on a FLASH chip on a Motherboard. In flash memory, the Intel specific pre OS FW can be flashed with variant partition size. And using SPI flash mechanism, BIOS and other peripheral FW can be flashed to the board.

3.2 IFWI

IFWI is Integrated FW Image binary which holds the absolute software of a system. The FW image contain the other images as the whole system have multiple components. So in addition to feature addition and loading the all FW component into IFWI Stitching is done.

Figure 3.2: IFWI Building

3.2.1 IFWI Stitching

Intel platforms utilize IFWI which have base IFWI and on top of it contains other firmware components to boot which are needed be signed and manifested. The signing of binary will ensure verified boot[7].

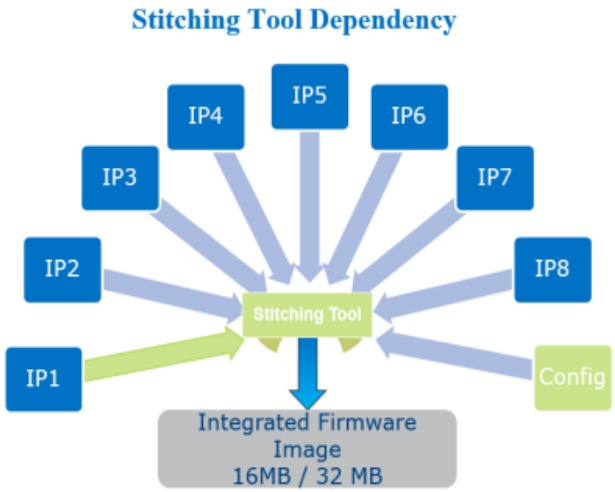


Figure 3.3: IFWI Stitching

Integration of all the Configuration of different IPs is done by invoking this tool. Manual process used to stitch one binary with different configuration files. There are different IPs and its Configuration which combined together by Stitching tool. At the End IFWI of 32/64 MB is resulted.

3.3 Failure Modes in safety System

The occurrence of Failures can be anytime in the entire system or in the implementation flow or in the design of a hardware component or software feature including the integration of all the components altogether in a system. Faults are classified as Systematic Failure or Random Failure.

Systematic faults which get inherited in the design and fall out in hardware, software and at the platform level resulting in the deficiency of the development, like:

- Incorrect or ambiguous specification
- Failure in the design and Validation process
- Failure in requirement understanding
- Tools or Binary generation failure

Random faults occurrence is during implementing and are originated due to manufacturing, ageing and conditional/situational based. Those failures are captured in hardware and arise disordering in software. It could be permanent or transient at the architecture level. It includes:

- Physical defects caused by Permanent Failures
- Environmental factors like radiation due to Transient Failures
- The statistical failure of asynchronous events due to Transient faults

3.4 Fault Detection Mechanism

Systematic Failure can be avoided by strong Verification strategy keeping Safety integrity level in reference. So the Failures can be surpassed using the stimulus-based testing Techniques which includes formal and semi-formal methods.

In Formal Methods proper use of mathematical approach such as Finite state machines and algebra which are implemented logically using specific tools including modelling languages which check the correctness and verify with the models.
Semi-formal specifications include pseudo-code and structured diagrams or models this decreases ambiguity in hardware and enables get an accurate result of the design.

A technique used is the Memory Management Unit (MMU) and also the Memory Protection Unit (MPU) to detect the imprecise data. Development of Software is done keeping hardware capability in reference and list out the Assumptions of Use (AoUs) and this development is specified in Hardware Software Interface (HSI).

Random faults are detected in hardware by software mechanism by employing nominal hardware. Random faults can be:

- Single Point Failure – one failure, like an open/short circuit
- Multiple point Failure – one or more failures, several simultaneous single point faults
- Latent faults – multiple point faults events that only occur under fault situation.

Once the indication of a fault is captured then diagnostic and corrective methods are undertaken either periodically or occasionally.

Diagnostic Types:

- Continuous Diagnostic
  - Temporal duplication with comparison of results known as Dual-Core
- Lockstep
  - Information surplus, such as Error Detection or cyclic redundancy check.
  - Analytical fault tolerance
  - Redundancy at the system level, known as Software Lockstep

- Functional testing using software (Software Test Library)
- Inspection based Built-In Self-Test (BIST) of logic or memories
- Demand-based test pattern checking

3.4.1 Information Redundancy

In this Technique is Correction Code and error Detection is included. Cyclic Redundancy Check is the best method to capture the error and it is highly effective way of detecting errors with bit cells within memories. A CRC generator logic is created inside the software at Host end to calculate the CRC for each cycle and send it to Intel Safety Island.

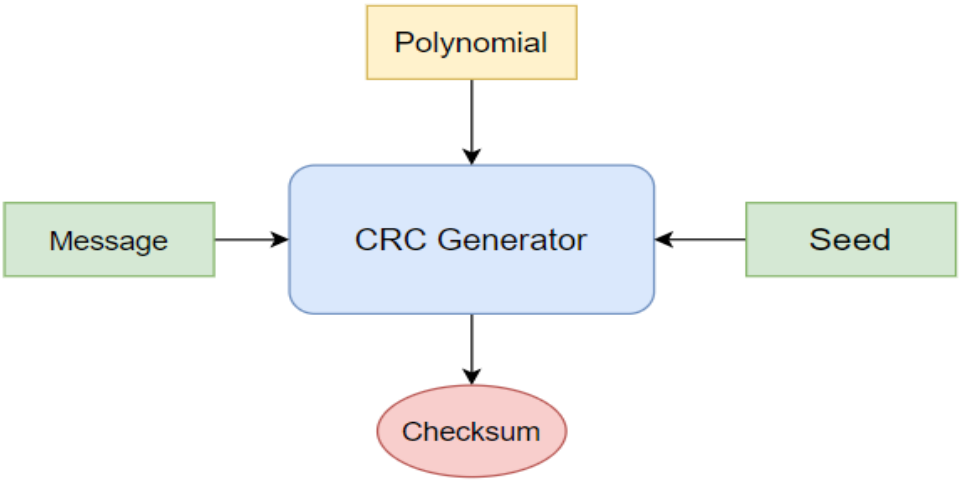


Figure 3.4: Cyclic Redundancy Check

In CRC detection method Polynomial is to be selected first it could be either 8,16,32 bit and Host will send the Message contacting the Data or Signature and Seed is to be added to it. An generating the CRC, a checksum is calculated out of it which indicates the presence of Error.

3.4.2 Software Test Library and Test patterns

Software Test Library is a Software-based Fault detection method were periodically checking is done using software logic within Framework Application. This will check the Hardware functionally by performing its test repetitively in a deterministic manner to target the maximum diagnostic analysis.

Software Test Library is also known as a self-test by software or sometimes Software BIST(SBIST). It will periodically test the processor along with the rest of the components/peripherals in a fixed short window. STLs operation is performed without hindering the system state and if any failure is observed then it can quickly stop the operation by generating an interrupt to the working system. STLs re-

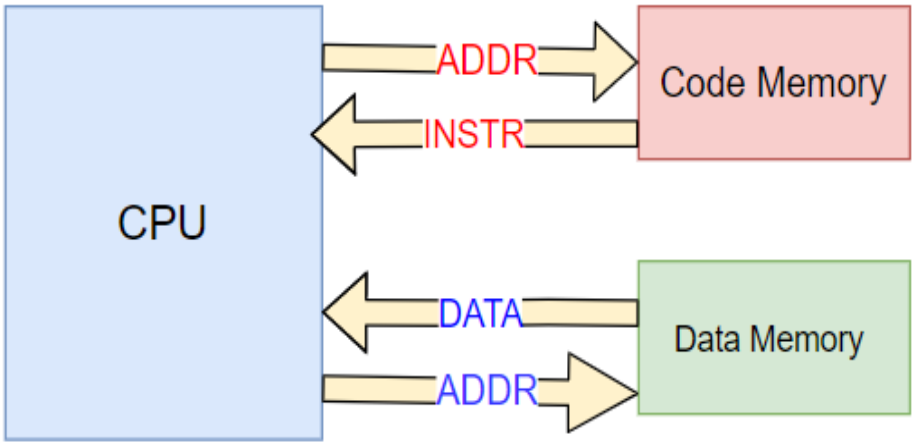


Figure 3.5: STL for a CPU-based system

quires less access to Hardware and require highest near to Host where the Customer Application Operating system is performed as it is in-field online testing carried out concurrently. STLs are executed in such a way to minimizes the disruption and interrupt service availability by tightly integrated with Real-time Operation system.

While system performing STLs, CPU continuously interplay with memory module for fetching instructions and reading and writing data to a particular memory register. During run time CPU also perform other safety and no safety-related work-

load which interdependently interact with other peripherals and self-test routine will induce a result to a single Register and create Signature. This Signature is checked by self-test code itself. If any error is captured due to generation of incorrect signature is flagged in memory Register with the cause of the failure. STL is performed

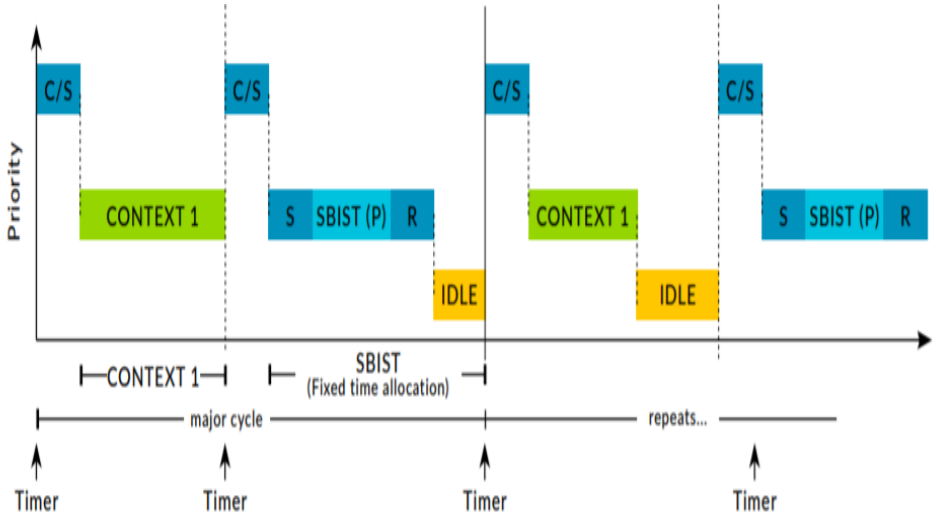


Figure 3.6: Software Test Library

on the basis of the regular dedicated interval time slice. It means during computation of a quantum, context switching will be prioritized and the task will be started on next quantum by fetching the previous data from the stored register and along with this it performs fetching of memory registers for reading/write operations as part of software BIST.

3.4.3 Lockstep

Processor lockstep is a method used to get the most stability in an Embedded system. Adding another identical processor in a system which monitors and cross verifies the activities undergoing in the system flow.

The two identical processors are started a the same time in system start-up mode, it will encounter the same feed in that can be either code, bus operations and

asynchronous events, so during normal operation, the condition of the two processors is alike from clock to clock. when this setup is working as expected then it is known as operation working under lockstep mode.

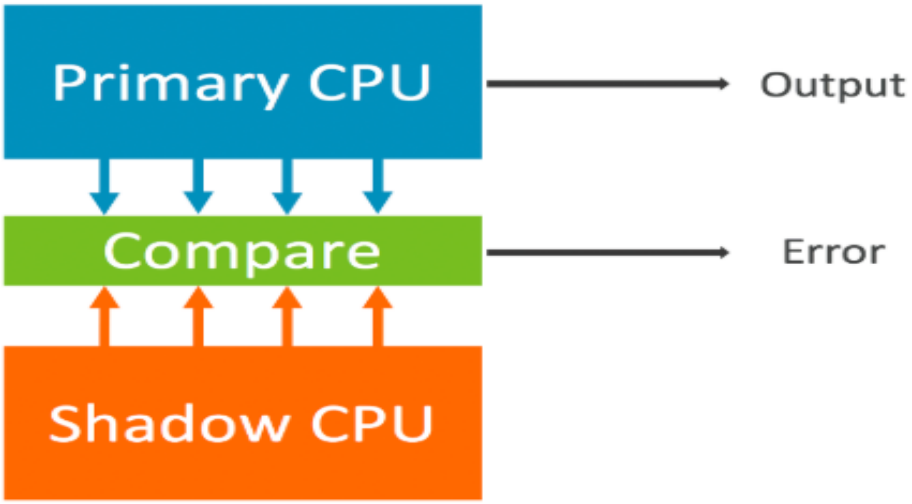


Figure 3.7: Lockstep Computing

Whenever an error is observed in any of the processors it will reflect a difference between the status of the two processors, which will eventually be evaluated as a difference in the outputs from the expected one, so the main aim of the lockstep computing is to monitor the outputs of the two processors and result from an error in the case of inconsistency or in-variance.

This method is aimed to detect the wide range of faults and also to achieve high diagnostic coverage but at the downside of identical Processor leads to increase in power cost and area cost.

3.4.4 Split Lock

The Split/Lock is the technique which empowers either Split mode or Lock mode.

In Lock mode, only the prime core is present functionally and the redundant core will be in the inactive state. That is the scenario when system boots or it in

reset mode in that state there is no memory sharing leading to increase the compute performance at lower diagnostic coverage.

In Split mode, both cores are active and logically present both cores work on independent clocks and have separate has its own independent clocks and the isolation of memory can handle the faults potentially.

3.5 Software based Lockstep Computing

Lockstep computing will require redundant processor which will mimic the original processor and safety related workloads will be performed but this will increase the hardware in terms of memory footprints. Since Intel CPUs do not have sufficient hardware safety mechanisms to meet the high level of safety standards. Typical Safety mechanisms like Hardware Lockstep used in the industry cause significant overhead in terms of SOC size and cost.

For an example Hardware Lockstep Safety Monitor – Dual-Core Lockstep This is used in different SOCs where Shadow core executes the same instructions as Main core and comparator compare the output cycle by cycle basis. The difference in outputs from the two processors will indicate the core fault. In the result of it is an application is effectively using two cores but only achieving the performance of a single core. This means that it leads to an expensive implementation. The main goal of software-based lockstep computing is to increase the efficiency without getting overhead of hardware by allowing the redundant core to execute the non safety-related application. This allows the cores to perform multiple Customer Workloads on the same cores along with the signature generation and its comparison.

3.6 Functional Safety Architecture

Safety Monitor component is in-charge of its own Functional safety in an Intel platform. It communicates with the Safety Supervisor and gives Status accordingly. So

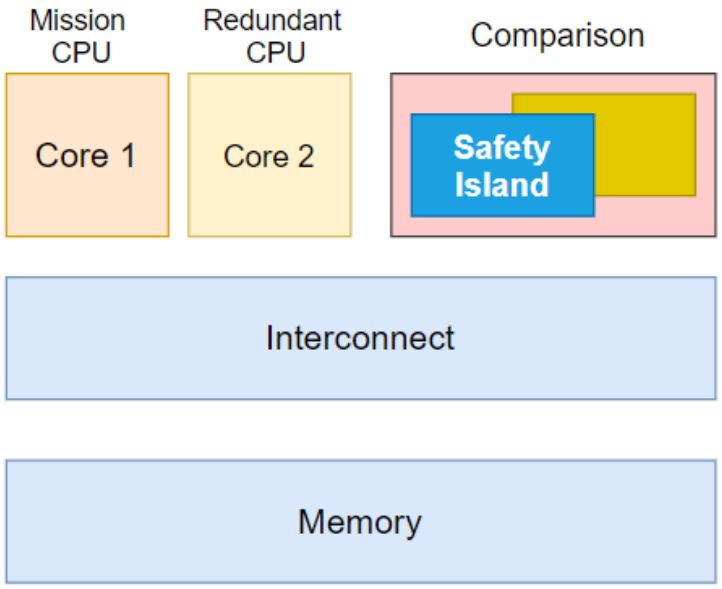


Figure 3.8: Software-based Lockstep Computing

it will work as Master and Slave.

Safety Monitor is a dedicated diagnostic IP that gathers all errors initiating from discrete blocks means from different parts of the SoC and routes to the system with dedicated pins.

The STL Manager will handle all the functionality related to STL execution on the host, Error logging for Intel Safety Island. The 4 cores will undergo the safety application or non safety-related operations. the Safety-related Application will be carried out on the basis of software-based Lockstep. It is responsible for software-based Lockstep services known as safety workloads. It provides APIs for starting and stopping the software-based Lockstep on the system. The Communication Library provides the services for communication with the ISI. The customer applications call the APIs in the user-space library for starting off the communication with the ISI. The Communication library will invoke the helper functions in the CRC generator for calculating CRC and for formatting the data required to be sent to the ISI FW. All the different software sub-components of the safety manager shall use the OS Abstraction Layer (OSAL) to isolate itself from OS-specific rou-

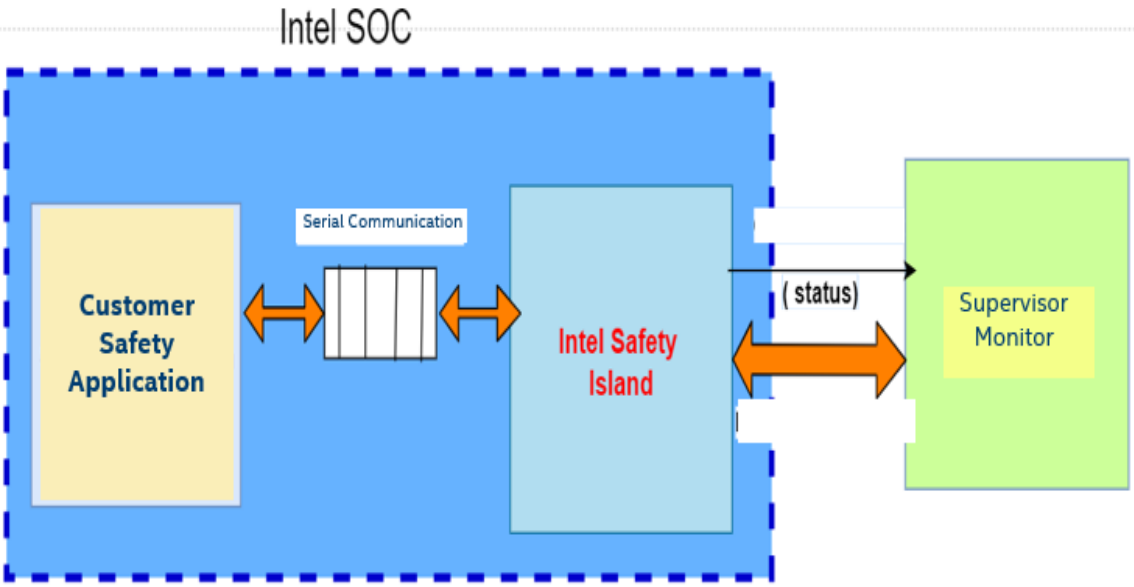


Figure 3.9: Functional Safety Architecture

tines. The communication with the ISI shall be a request-response form with Host. The STL manager and the software-based Lockstep manager can as well be developed independently by the customer. The STL manager and the software-based Lockstep manager are reusable as these software components are developed in "C" and they call the OSAL API for OS independence. The Communication Library and software-based Lockstep functions also are developed on OSAL API for OS independence. The host driver is the PCI to communicate with bus driver supported on the platform. The host driver also is attached to the OS framework and hence has some OS level dependence.

An OK/NOK/Alert signal indicates the overall functional status of the ISI CPU chips and these signals may be connected to an MCU on the platform or may drive out some relay switches

Safety Island also performs the role of a diagnostic microprocessor and monitors the health of the SOC in the platform. Safety island controls and takes part in Safety Mechanisms execution to detect transient and permanent faults, for example, it executes STLs (Software Test Libraries). Safety Island also acts as a Central

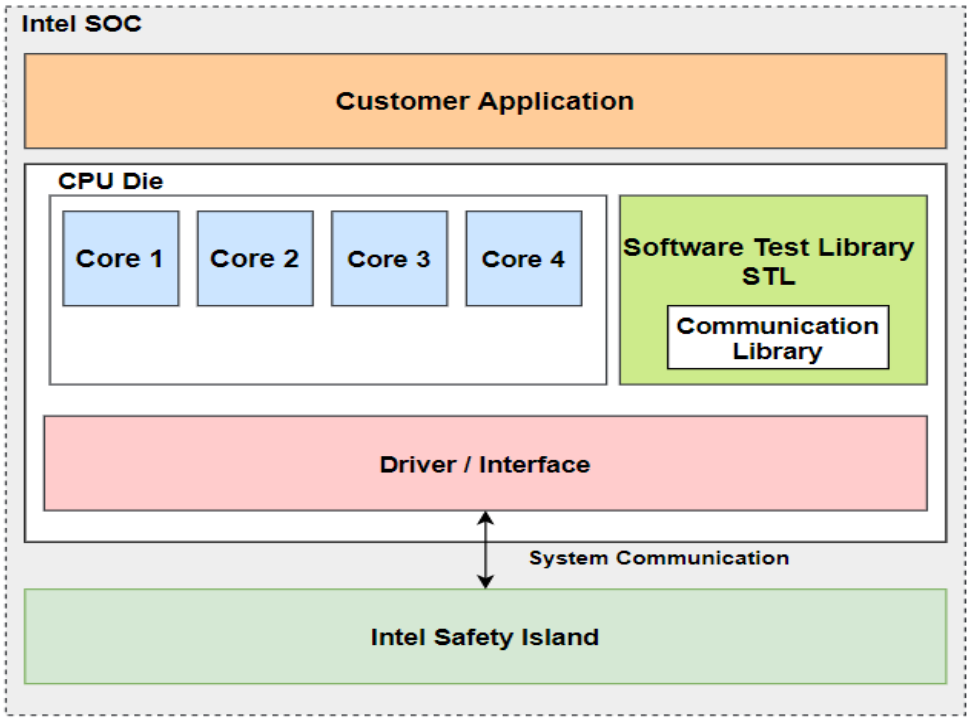


Figure 3.10: SOC Block Diagram

Alarm Collector for all the SOC safety alarms and reports them to the Safety Island FW stack and to an external MCU or optionally performs actions based on user configuration.

3.7 Voting Systems

Voting System is playing a critical role in Industrial and Automation Control where it helps in notifies alert and interlock the system when any failure is observed in the system. This will happen when the ongoing running Test Result fails and the actions are taken according to the system structure. Such System will help in executing safety functions to avoid the occurrence of any accidents.

So in simple terms, the system will have the sensor, input circuit, output circuit, processing unit and executing unit. In the presence of Diagnostic circuit with multi-

channel, it will behave like feedback information to the system which is used to do self-diagnosis of the system while saving the HW cost and in meantime also performing actively.

In case of one out of one system (1oo1) is insecure system where in case of failure, system will be stopped without any prior redundancy check or failure state check.

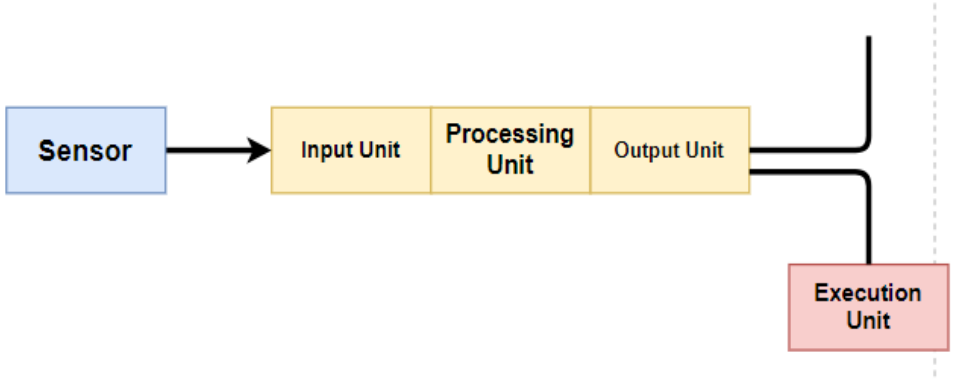


Figure 3.11: One out of one System Structure

In case of one out of two system (1oo2) there are two independent subsystems which have two output which decreases the possibility of system failure as one of the system will be in active state but this system is not able to diagnosis in case of Failure.

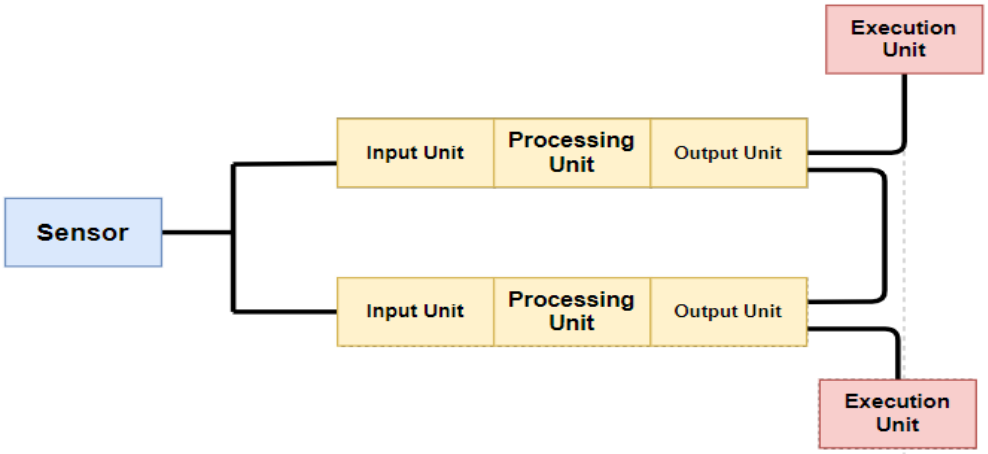


Figure 3.12: One out of Two System Structure

In Case of one out of two with Diagnostic (1oo2D) system will have its own diagnostic circuit connected to a system wherein case of any failure in result it will alarm its own system and another system about the failure and it will safely break the connection and work as 1oo1 System. It will detect the dangerous failures and allow actuators to be in a safe state.

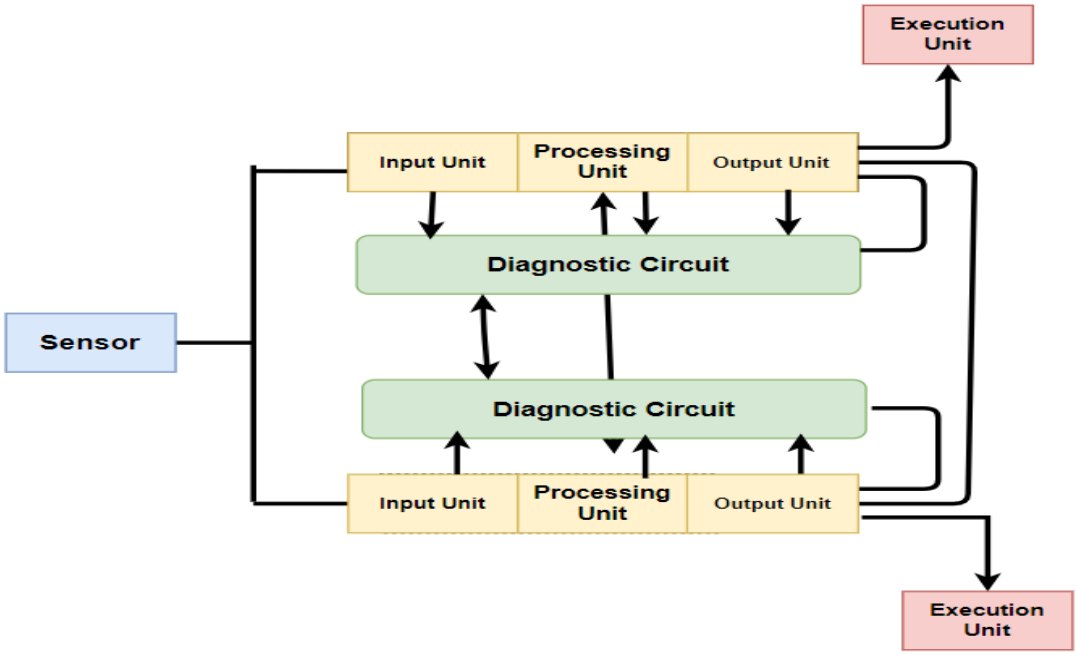


Figure 3.13: One out of Two Diagnostic System Structure

3.8 Proof Testing

Proof testing is explained in IEC61508 as a 'Periodic test executed to detect dangerous hidden failures in a safety-related systems, if necessary, a repair can restore the system to an "as new" condition or as close as practical to this condition'. A proof testing purpose id to identify the undetected failures which the system is unknown. It is an integral subsystem of the maintenance of the Intel safety SOC. A proof test is done periodically to identify dangerous failures, test safety-related functionality

for example reset, bypasses, alarms, diagnostics, manual shutdown, etc. The results of proof testing are also a measure of the effectiveness of the SIS mechanical integrity system and its reliability.

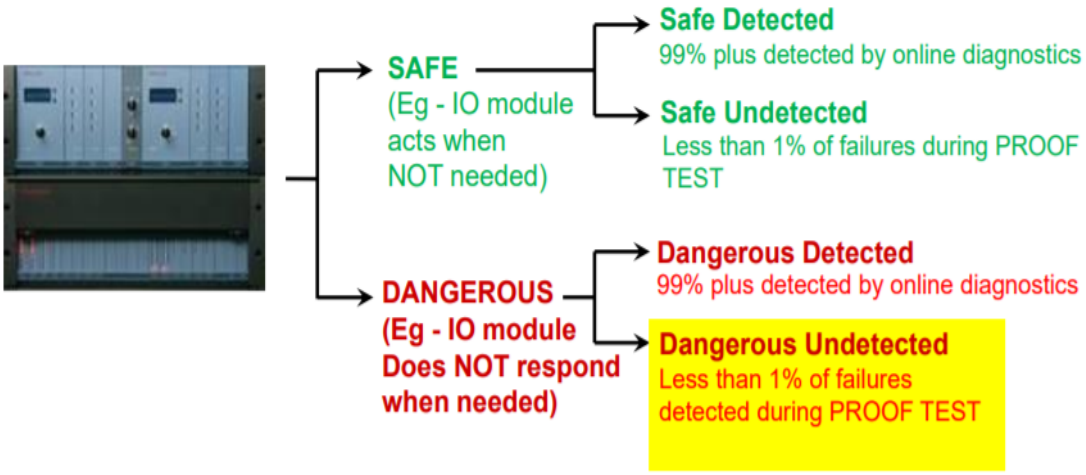


Figure 3.14: Proof testing

Chapter 4

Software Validation Techniques

In this chapter, the Validation scope, mythologies and testing flow are explained as part of Validating the Functional safety module. The Development of Test cases using Inter-Process Communication

4.1 Software Development and Testing

Software Development Life-Cycle is the sequence of events performed out by Developers to design and develop high-quality reliable software.

SDLC involves the coding tasks of the Developers as well as Validators who contribute their involvement in making Software productive. Third-party involvement is also there were stack-holder customize the software according to their requirement.

Software Testing Life Cycle involves Validation and Verification which contains series of activities carried out with Validation Techniques to test software product. STLC also involves Validator and Developers to interact with different modules and verifies the program flow in all possible ways. Test cases are planned in earlier basis and then Test Cases are Created depending on the feature.

4.2 Validation Techniques

4.2.1 Waterfall Model

Waterfall model is a serial model split into layered phases of software development activity. Where each stage has specific activity and task to perform during the SDLC phase. Validation phase in waterfall model starts once the implementation of the system is done.

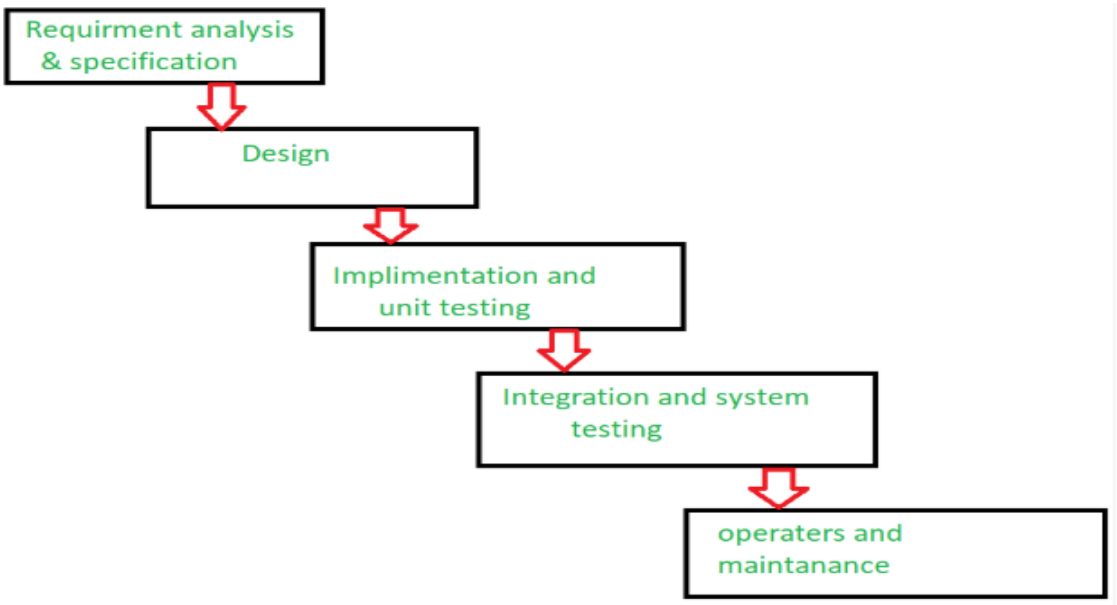


Figure 4.1: Waterfall Model

There are few limitations listed below for the waterfall model[4].

- One way street: After the end of the first phase, the next phase starts then there is no other way to go reverse on the last phase. It is one of the failures of the waterfall model.
- Interaction: Lacked interaction with each other phases. A development process begins, changes are not accepted easily.
- Support delivery of system: The waterfall model does not allow distribution of

the system in pieces. Once the development process initiated, changes cannot be implemented easily.

- Feedback path: This model does not support any feedback path. The waterfall model thinks that no error is ever carried out by developers during any phases. Hence, no error correction is required.

4.2.2 V - Model

The V-model is an SDLC model in which operation of processes occur in a sequential fashion in a V-shaped . It is described as Verification and Validation model[5]. It is constructed on the grouping of a testing stage for respective development phase. Development is directly co-related with its testing. The further Step imitates only after previous stage get finish. So Validation and Development runs parallelly in this model.

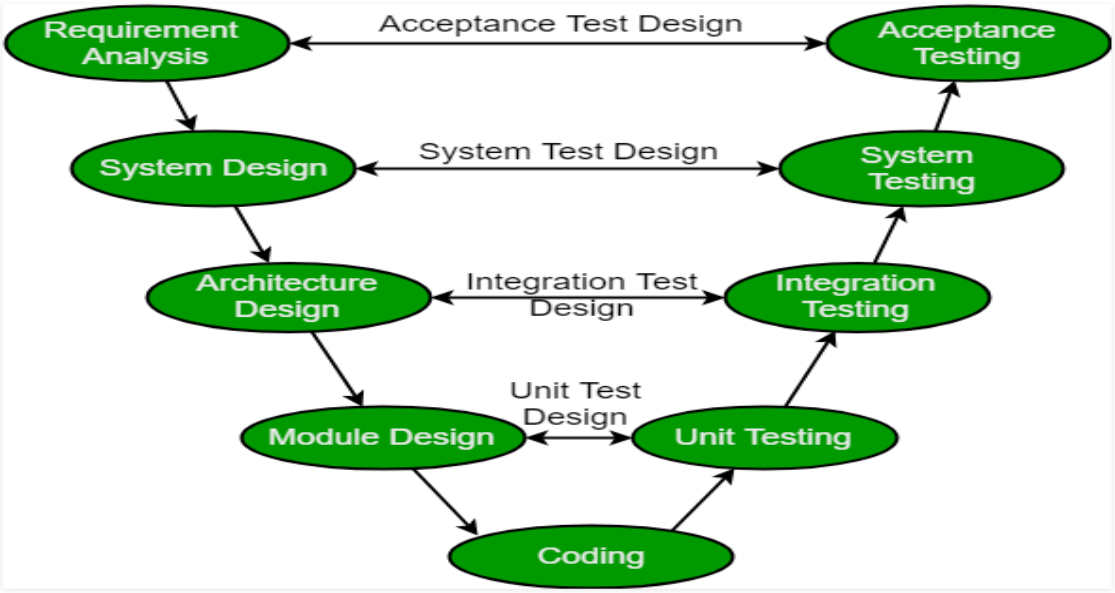


Figure 4.2: V-Model

V-model has two partitions where one part is Development and another part is Validation. That is the reason it is named as V-model.

In Design Phase there are sub-parts:

- **Requirement Analysis**: This phase gathers the requirement and expectations of Customer in Detail and further work accordingly.
- **System Design:** This Phase has complete information about hardware setup and Develop a System.
- **Architectural Design:** This phase has the broken down modular and functional information about each module and also have inter-module connection understanding. In-short have an overview and Detailed Structural Data within it.
- **Module Design:** This phase further split the system into a module and particularly feature along with detailed Design Knowledge of each module.

In Testing Phase there are sub-parts:

- **Unit Testing:** This phase will do unit level testing to eliminate bugs and issues in the code of Design Team.
- **Integration testing:** This Phase has complete information about hardware setup and Develop a System.
- **Architectural Design:** This phase verifies the communication among the modules by integrating them at the same level so that it can verify Architecture Design.
- **System Testing:** This phase will verify the whole system with its expected functional behaviour with its inter-dependencies.
- **User Acceptance Testing (UAT):** This is the ultimate Stage where all user requirements are met as the system is ready to be launch in the real world. UAT is carried out in the user environment which is relevant to the production environment.

Concept of V-Model:

- **Hierarchical Approach:** Each phase gives more and more detailed information about the Design.
- **Scalability:** V- model is Flexible to follow and customer product requirements are identified clearly.
- **Cross Referencing:** It gives Direct Correlation between Development and Validation- Verification task.
- **Systematic Documentation:** Documentation is done by both the Design and Support team to maintain the system productively.

4.3 Inter Process Communication

In a safety-critical system, a real-time operating system (RTOS) is used where the component sustains the verification. For aimed safety standard its necessary to follow the regularity certification by following the proper process of testing. In automotive, industrial, railway and medical applications FuSA RTOS is used to enable and for the smooth run of the product safety.

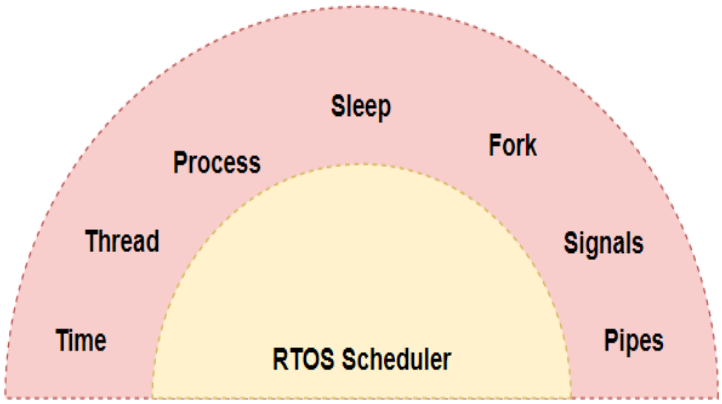


Figure 4.3: RTOS

FuSa RTOS is a deterministic real-time operating system which accurately controls multiple threads of applications with a different priority, and follows preemptive scheduling. It provides services that are required in composite real-time applications, such as threads, processes, files, signalling, timers, memory and object management, message exchange and others.

As part of Validation process created the Test cases which communicate with the other processes which are targeted to run on different cores. this communication between different processes and threads is done by Inter-process communication using Signalling and pipes. The scheduling the processes in Linux is done by the scheduler and its library. Core affinity used for dedicating the particular process to be executed on a particular Core in CPU die. The wide use of Fork is carried out for all the Test Cases for dedicated parent and child process.

Chapter 5

Results and Outcomes

In this chapter, as part of Validation of functional safety module certain process is followed to develop test cases and to execute the test cases. This chapter gives detailed view of Validation cycle, Test Development, Test coverage and Test results.

5.1 Phase I: Validation Cycle

The whole system has a specific predefined methodology to make the system valuable. And that is started with Test planning of features which contains the detailed analysis and working scenarios and then it gets reviewed by Senior members and Architect.

The Test plan of Functional Safety Module is carried out with in depth study of following specifications,

- Hardware Architectural Specification
- Software Architectural Specification
- Modular Architectural Specification
- ISI Diagnostic Specification
- Customer Requirements

After that Test case Development phase starts where from a validation point of view Test are created including all the positive-negative flows. Those tests are executed to validate the Firmware released by Developers and report bugs accordingly. During this phase Debugging and Code change is performed and in new Release, those fixes are included. Again the same process repeats to make the system bug free and get a productive outcome.

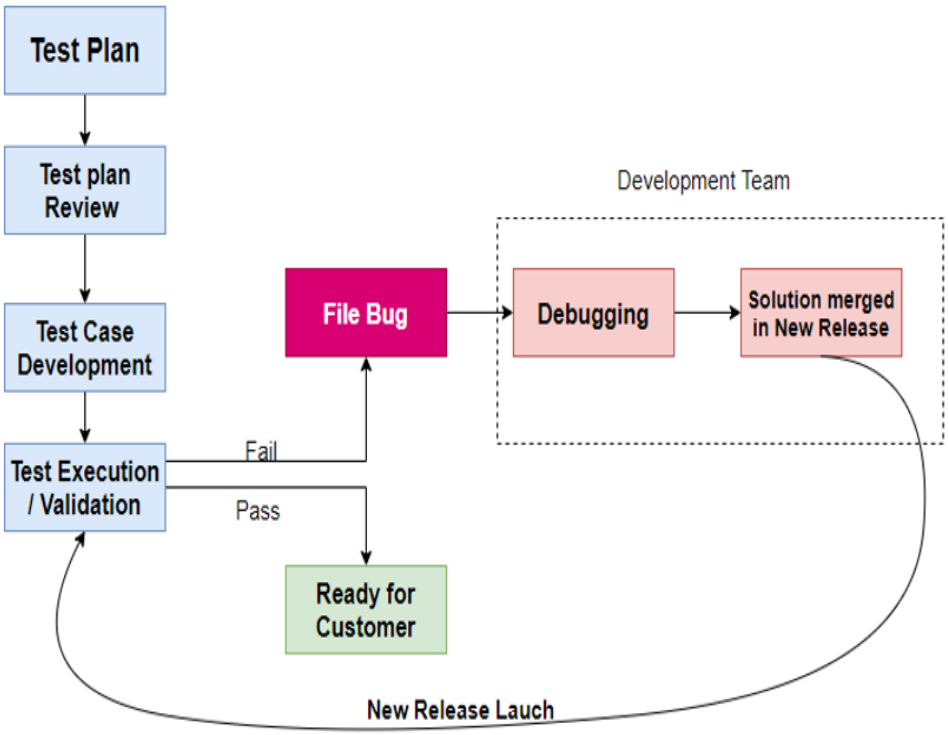


Figure 5.1: Validation Process

5.1.1 Test Case Development

The organised Validation process is designed and it is in synchronisation with Development. Initially, Test plan has to be created depending on all architectural and functional flows. Those Test Plans gets reviewed by higher management and after this process, Test Development phase starts where all positive, negative and corner

cases are kept in consideration meeting the functional requirement. Then execution phase starts where the binary is created, containing all different IP's binary and it also contains ISI Firmware binary. on Flashing that binary on Silicon board will boot up and on to that Validation Test are executed. On failing Test, debugging phase starts and the report is sent to the Development Team. So, in Next Firmware Release fixes for the bugs are provided. and this cycle goes on enabling the predefined features and meeting all Customer based requirements.

Across all the different Configuration, Developed around 61 Test cases with a variety of Combinations to cover positive flow, Negative flow and the corner cases to validate the ISI Firmware. Ensured that the Customer Requirements are met by validating the features assigned.

Table 5.1: Test Count

Sr no.	**Test case Development**	**Test Count**
1	Unit Test	32
2	Functional Test	10
3	Negative Test	9
4	Stress Test	4
5	Integration Test	6
	Total	61

Host side Test Case Development in Linux using Inter-process Communication and scheduling of the process.

Firmware is side-loaded in Kiel IDE for debugging the Firmware execution by checking the memory and watch windows in IDE.

5.1.2 Coverage Details

To measure the effectiveness and the potency of the Test or the Code for specific feature, Test Coverage and Code coverage mythologies are used.

In simplest language the Code coverage means the percentage of Test that is enfolded by the Code either manually or by automating the testing and using any

Test Framework. coverage sum up the testing of the features executed as a part of the Functional based requirements specification, software-based requirements specification, and other required entities.

Table 5.2: Test Coverage Details

Sr no.	**Test Coverage**	Percentage
I	**Integration Test**	
a.	Functional Coverage	77
b.	Call Coverage	82
II	**Unit Test**	
a.	Statement Coverage	95
b.	Branch Coverage	91
c.	Branch Coverage	71

5.2 Functional Safety Modules

The functional safety modules in Intel SOC is targeted to detect the fault occurrence and detection. For validation the Functional safety modules, following features are owned,

a. Communication Libraries

b. Software Test Libraries

c. Software based Lockstep in 1oo1 and 1oo2D systems

d. Proof Testing

e. Host to Host Services

5.2.1 Communication Libraries

The main focus during Validation phase was on the communication between the Host to Intel Safety Island. The Communication Library provides the services for

communication with the ISI. The customer applications call the APIs in the user-space library for starting off the communication with the ISI. The Communication library will invoke the helper functions in the CRC generator for calculating CRC and for formatting the data required to be sent to the ISI FW. To validate the communication, use of APIs are done and Test cases are develop to verify the APIs where each APIs have its own functionality.

5.2.2 Software Test Libraries

The Software Test Libraries will be provided as a library to the customer to be integrated to customer application. The library will contain functions to be invoked during startup. STLs will periodically test the processor along with the rest of the components/peripherals in a fixed short window. STLs operation is performed without hindering the system state and if any failure is observed then it can quickly stop the operation by generating an interrupt to the working system. Validation is done by sending the pair of correct STL number and incorrect numbers to verify the validity of STLs results.

Software based Lockstep in 1oo1 and 1oo2D systems Validation of the Software based Lockstep in 1oo1 system is done by scheduling the processes to be run on different cores and the comparison of output is done within 2 cores and output is created on basis of the pass or fail output. Validation of the Software based Lockstep in 1oo2D system is done by scheduling the processes to be run on different channels of different Systems having different ISI and the comparison of output is done from different channels and output is created on basis of the pass or fail output. The Scheduling of the processes is done by the using the OS features i.e. forking the parent and child processes, scheduling the process to particular core, using Inter process communication to communicate or to pass the signals to two processes.

5.2.3 Proof Testing

Proof testing includes action from getting permissions, generating notifications and moving the system out of action for testing to confirm inclusive testing, registering the proof testing and their results, keeping the system back in action, and comparing the current test results and past proof test results. Validation of proof test is done by creating the Binary with correct Proof testing content and then on flashing the correct binary with FW code, Proof testing is performed which gives out the results on basis of the performed configuration of registers.

5.2.4 Host to Host Communication

Host to Host Communication is carried out in 1oo2D system where the two Intel SOC will be forming its own tasks and in case of any failure, systems collects the Diagnostic Data and notifies the other Intel SOC about the failure and release its connection and turn out the functionality into 1oo1 system. Test cases are developed to generate the failure scenarios and inject the error to make the 1oo2D system to end working as 1oo1 system after detecting the failure in one of the Intel SOC.

5.3 Phase II: Basic Acceptance Test

Basic Acceptance Test is set of basic functionality test which are performed during every ISI Release done by development team. The Validation of BAT covers the basic mandatory tests which enables the basic functional test of the all the IPs within ISI is performed. On every Release the perfect set of Binaries are created having all the components/IPs enabled and stitched together to make one valid binary which contains the ISI FW image along with the other important Components all over the Intel SOC.

Chapter 6

Use Case

As a safety compliant component, the Intel SoC can be used in a wide variety of industrial applications which demand safe operation. This section describes this general use case and the supporting safety functions, and functional safety requirements.

6.1 Industrial Use Case

The general use case for lake series of Intel is the "control logic" shown in Figure 5.1 and its main functions are:

* Receive input from input devices (sensors, switches, counters etc.);

* Process input received from input devices as demanded by application software;

* Transmit processed data to output devices or other interfaces as demanded by application software for normal operation and under error conditions. As shown in Figure 5.1 several input devices (sensors) are processed by the control logic and some output is driven in response.

There is a display, a Human Machine Interface (HMI) with associated software, and some memory storage with its own software.

The HMI interface and display are assumed not to contribute to safety function.

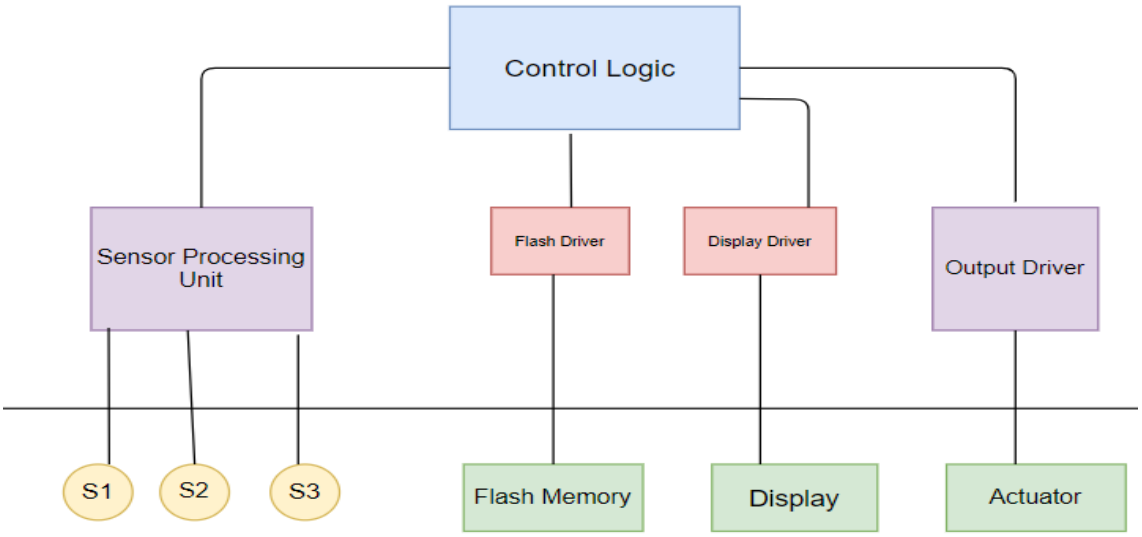


Figure 6.1: Industrial Assumed Use Case

Input devices: Input devices are used to obtain information about the system (machine). Input devices can be a variety of device types like sensors, switches, buttons, counters etc. The input information depends on the application need and can be among the following: process parameters, machine health parameters, user interactions, another machine or programs interactions, environmental parameters etc.

Control Logic: The Intel SoC implements the control logic. Intel SOC receives process input from input devices and executes the safety control algorithm as demanded by application software. The processed data is then analyzed by the control logic for actions that are commanded to the output devices.

Output devices: The output devices again can be a variety of devices like relays, solenoids, motors, digital modules etc.

Intel SOC is to address an industrial automation with the emergence of Industry 4.0, where the evolution emerged across the architecture. This change is mainly evolved by the necessity to connect the machines to a common cloud infrastructure, gather data from these machines constantly to understand and improve their effi-

ciency, control them in a time-sensitive way to avoid unnecessary cycles and localize the intelligence within the machines to drive better resource allocation. Customers are looking at enabling these systems like programmable controllers, Robotic Armour motor/motion controllers to meet their goals of efficiency.

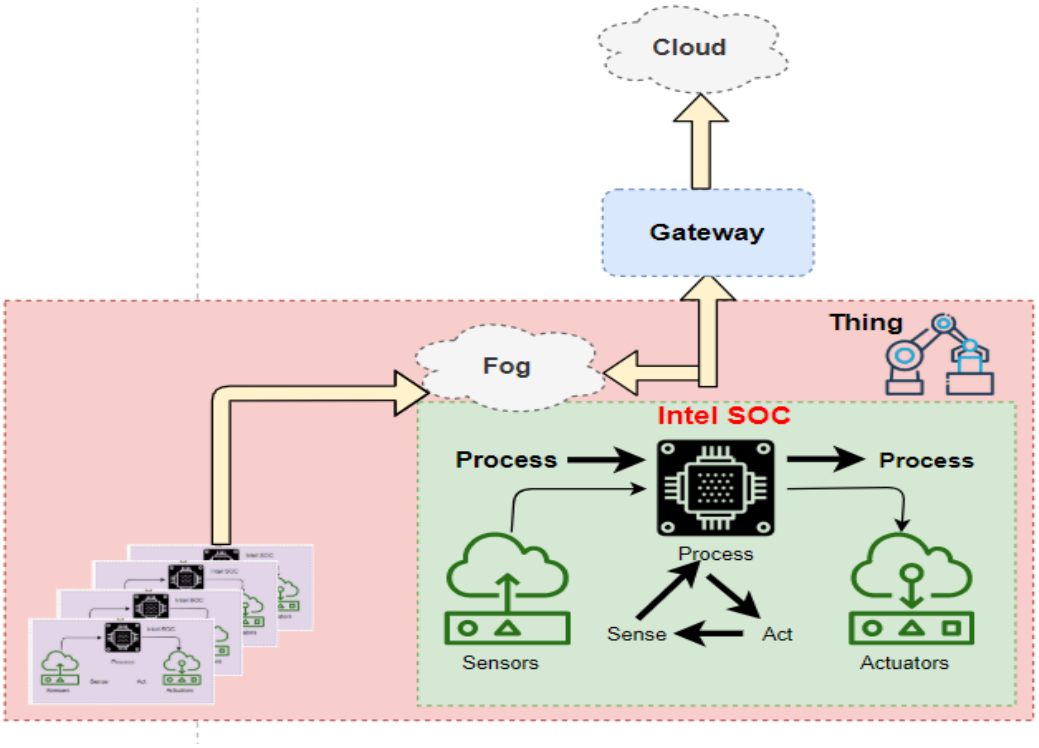


Figure 6.2: Block Diagram of IoT in Industrial Application

A majority of control systems being managed at the edge to beat the round-trip latency to cloud in mission-critical applications, there is always a need to improve the compute performance at the edge. Systems that can be upgraded to newer high-end MPUs for better Networking.

An IoT gateway used as a data aggregator with inputs from several edge sensors or devices and is expected to do a first level analysis/compute before pushing the data either to cloud or back to the edge devices with appropriate actions. A gateway can be implemented as a SW or a HW solution. Intel SOC is expected to be a standard compute hub with multiple IOs connected to it. The sensor or edge device

data can be transferred over any of the high-speed IOs. The edge device can be an industrial machine connected over a standard Ethernet connection smart-city hub which is expected to collect continuous data from multiple different sensors in a given area. The data collected across all the possible IO connections to Intel SOC is expected to be processed on any of the processor subsystems within the SoC. There is a wireless connection to the cloud is created to support one of the desired cloud management SW for customer implementation.

Chapter 7

Conclusion

In this thesis, Functional Safety is aimed for the domain of Industrial Application. The target for the project work is to provide Strength to the creator of functional safety systems. As the goal of this project is enable the FuSa Element Intel SOC for the safe functioning of the System.The construction of functional safety systems is dependent on standards. The standard IEC 61508 list out requirements and recommendations of a customer by following the Design and Technical flows, its architecture and methods to be applied in the Development and Validation Stages of V-model in Functional Safety Systems. The Validation of modules is to be performed by looking into its basic Functionality and its behaviour on Customer Platform and also on Hybrid FPGA.

In process of completion of this Project, the generation of different test cases is carried out. These test cases were based on the requirement of the Customer and were developed by considering the stability of the module. Different timing critical parameters are considered in scheduling the threads on different cores for the generation of fully functional test cases to get validate the Intel Safety Island Firmware. Positive, negative, Stress and Corner Scenarios are covered in the Validation phase to make the Firmware of the Functional Safety element robust.

The main focus during Validation phase was on the communication between the Host to Intel Safety Island where it check for software based Lockstep, Software

Tested Libraries, Host to Host Communication,

The conclusion of this project can be drawn to get functional safety function in a system to process the feedback from the diagnostic circuits either from same Intel Safety Island or from different channels of same Intel Safety Island for getting information about certainty and uncertainties. The presence of the self-diagnostic Circuit will help to decrease the possibility of sudden system failure.

7.1 Future Work

Optimizing the product by using the same or common artifacts and pass out the higher payoff. Aiming for Validation of Functional safety module with optimal effort and optimal configuration. In future scope to reduce the Hardware and to improve the reliability of diagnostics by making the system as one out of three Diagnostics where three safety systems are connected to each other and in case of any fault, any one of the systems goes down and other two systems continue its functionality. This will ensure the system can be more efficient and can be more robust to any failure. The Module needs to show applicability and feasibility in more advanced Industrial Cases which reduces the failure or fault occurring ratio.

References

[1] "IEC - International Electrotechnical Commission." Www.Iec.Ch, 2010, www.iec.ch/functionalsafety/?ref=extfooter. Accessed 14 Dec. 2019.

[2] Jari Rauhamäki , Designing Functional Safety Systems: A Pattern Language Approach. (Publication; Vol. 1478), Tampere University of Technology, 2017, TUTCRIS Portal "https://tutcris.tut.fi/portal/files/11033872/rauham$_k i_1$478.*pdf*"

[3] Stephan Baumgart, INCORPORATING FUNCTIONAL SAFETY IN MODEL BASED DEVELOPMENT OF PRODUCT LINES, Mälardalen University Press Licentiate Theses, 2016. "https://www.diva-portal.org/smash/get/diva2:906392/FULLTEXT02.pdf'

[4] S. Chonnad, R. Iacob and V. Litovtchenko, "A Quantitative Approach to SoC Functional Safety Analysis," 2018 31st IEEE International System-on-Chip Conference (SOCC), Arlington, VA, 2018 S. Chonnad, R. Iacob and V. Litovtchenko, "A Quantitative Approach to SoC Functional Safety Analysis," 2018 31st IEEE International System-on-Chip Conference (SOCC), Arlington, VA, 2018.

[5] Liu, Bohan, et al. "An Incremental V-Model Process for Automotive Development." 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), 2016, ieeexplore.ieee.org/stamp/stamp.jsp/tp=arnumber=7890592, 10.1109/apsec.2016.040

[6] International Electrotechnical Commission. IEC 61508. Functional Safety of Electrical/Electronic/ Programmable Electronic Safety Related Systems. Geneva: IEC Press; 2000.

[7] "Stitch IFWI Image — Slim Bootloader 1.0 Documentation." Github.Io, 2018, slimbootloader.github.io/developer-guides/stitching-ifwi.html. Accessed 14 Dec. 2019.

[8] Community.arm.com. 2020. Comparing Lock-Step, Redundant Execution Split-Lock. [online] Available at: "https://community.arm.com/developer/ip-products/system/b/embedded-blog/posts/comparing-lock-step-redundant-execution-versus-split-lock-technologies" [Accessed 9 May 2020].

[9] S. K. Das, "Design and Modelling of Smart Environments: A Framework based on Learning and Predictio," 2007 2nd IEEE Conference on Industrial Electronics and Applications, Harbin, 2007

[10] Xi, Y., Liu, F., Yuan, H. and Pan, D., 2013. Safety Voting System Based on D-S Evidence Theory. TELKOMNIKA Indonesian Journal of Electrical Engineering, 11(

[11] O'Reilly Online Learning.2020.Interprocess Communications In Linux: The Nooks Crannies. [online] Available at: "https://www.oreilly.com/library/view/interprocess-communications-in/0130460427/" [Accessed 9 May 2020].

[12] 2020. [online] Available at: "https://www.miinet.com/food-and-beverage/alarm-trip-1-out-of-2-voting-with-high-availability" [Accessed 9 May 2020].

[13] Kenexis.com. 2020. [online] Available at: "http://www.kenexis.com/wp-content/uploads/2018/11/Comparison-of-2oo3-voting-and-2oo2-voting-1.pdf" [Accessed 9 May 2020]"

[14] GUO Haitao, YANG Xianhui. Quantitative Reliability Assessment for Safety Related Systems Using Markov Models. Journal Tsinghua Univ (Sci Tech). 2008; 48: 149-152,156.

[15] F. Kastensmidt and P. Rech, FPGAs and Parallel Architectures for Aerospace Applications: Soft Errors and Fault-Tolerant Design. Springer, 2016.

[16] M. Kooli, P. Benoit, G. Di Natale, L. Torres, and V. Sieh, "Fault injection tools based on virtual machines," in Reconfigurable and CommunicationCentric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on. IEEE, 2014.

[17] A. Velasco, B. Montruccio, and M. Rebaudengo, "A hardening approach for the scheduler's kernel data structures," in CompSpace at ARCS2017, 2017.