Zynq UltraScale+ MPSoC Safety Manual

User Guide

UG1226 (v1.1) August 21, 2017

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/21/2017	1.1	Added notes about the FMEDA Report and Temperature Profile under Overview in Chapter 5.
07/21/2017	1.0	Initial Xilinx release.

Table of Contents

Introduction

Overview

This manual is part of the safety documentation related to the Xilinx® Zynq® UltraScale+™ MPSoC and its purpose is to describe the use of the Zynq UltraScale+ MPSoC device in the context of a safety-related system, specifying user responsibilities for installation and operation of these devices in your safety systems to maintain the desired safety integrity level.

This safety manual is written in compliance with the Automotive Safety Standard ISO-26262, edition 2011/2012, also taking in account the guidance given by the Draft version of ISO-26262 Part 11 available in April 2016 (Part 11, together with the complete Revision 2 of ISO-26262 will be officially published in 2018).

The general assumptions of this safety manual are that the item integrator will implement all of the requirements pertinent to the application including implementation of the assumptions of use, inclusion of the software test libraries described in the *Zynq UltraScale+ MPSoC Software Safety User Guide* (UG1220) [Ref 8] and enablement of all hardware safety mechanisms. Through application of the relevant requirements, the Zynq UltraScale+ MPSoC device can potentially support safety levels up to and including SIL3 and ASIL-C (see Table 4-1: Safety Configurations). Deliberate non-application of any requirement is acceptable provided the safety goal of the item is achieved. This may occur when the safety goal has a safety integrity level lower than ASIL-C or the safety goal is achieved through different measures at the system level, or a combination of both. Furthermore, if any requirement stated in this Safety Manual is not fulfilled deliberately by the Item integrator a rationale shall be provided that motivates the deviation. The rationale shall be consistent with the achievement of the Item Safety Goal, and will be considered acceptable as long as they are compliant to ISO 26262-10:2011, Clause 9.2.2 (see Appendix C).

Reference Documents

Appendix D, Additional Resources and Legal Notices identifies all documentation referenced in this safety manual.

Safety-Related Deliverables

The majority of safety-specific material listed below is available for registered users to access via the *Xilinx Functional Safety Lounge* (https://www.xilinx.com/support/quality.html). This lounge also provides references and links that have broader appeal such as the *Isolation Design Flow* which is applicable to safety and security -focused designs (https://www.xilinx.com/applications/isolation-design-flow.html).

1. *Zynq UltraScale+ MPSoC Safety Manual* (UG1226)
2. *Zynq UltraScale+ MPSoC Software Safety User Guide* (UG1220)
3. Zynq UltraScale+ Software Test Libraries (STL) Source
4. Zynq UltraScale+ MPSoC FMEDA Report
5. *Vivado Safety Manual* (UG1187)
6. *Xilinx Quality Manual* (QAP0002)
7. *Device Reliability Report* (UG116)
8. *Isolation Design Flow for Xilinx 7Series FPGAs or Zynq-7000 AP SoCs (Vivado Tools)* (XAPP1222)
9. Soft Error Mitigation (SEM) Core

Acronyms

This section identifies all acronyms used in this safety manual.

ADAS	Advanced Driver Assistance Systems
AEB	Automated Emergency Braking
AEC	Automotive Electronics Council
AoU	Assumption of Use
APU	Application Processing Unit
ASIL	Automotive Safety Integrity Level
AXI	Advanced eXtensible Interface
BBRAM	Battery Backed Random Access Memory
CAN	Controller Area Network
CCF	Common Cause of Failure
CIB	Cryptographic Interface Block

CLKMON	Clock Monitor
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRL	Clock and Reset Logic
CSU	Configuration and Security Unit
DAP	Debug Access Port
DDR	Double Data Rate (Memory)
DDRC	DDR Controller
DFA	Dependent Fault Analysis
DFMEA	Design Failure Mode Effect Analysis
DMA	Direct Memory Access
DSP	Digital Signal Processing
ECC	Error Correcting Code
ECU	Electronic Control Unit
E/E	End to End (protocol)
FCI	Flow-Control Interface
FFI	Freedom From Interference
FIFO	First In First Out (buffer)
FIT	Failure In Time
FMEA	Failure Mode Effect Analysis
FMEDA	Failure Mode Effect and Diagnostic Analysis
FPD	Full Power Domain
FSBL	First Stage Boot Loader
FTTI	Fault Tolerant Time Interval
GEM	Gigabit Ethernet MAC
GIC	Generic Interrupt Controller
GPIO	General Purpose Input/Output
HFT	Hardware Fault Tolerance
HPC	High Performance Coherent I/O
HPM	High Performance Master
HW	Hardware
IDC	IDentity Compressor
IEC	International Electrotechnical Commission
IOU	Input Output Unit
IRO	Internal Ring Oscillator
ISO	International Organisation for Standardisation
I2C	Inter-Integrated Circuit (bus)

JTAG	Joint Test Action Group
LBIST	Logic Built In Self-Test
LFM	Latent Fault Metric
LLPP	Low Latency Peripheral Port
LPD	Low Power Domain
MBIST	Memory Built-In Self Test
MPFDI	Multiple Point Fault Detection Interval
MCU	MicroController Unit
MPSoC	Multi-Processor System on Chip
MPU	Memory Protection Unit
MRD	Market Requirements Document
OCM	On-Chip Memory
OCV	On-Chip Variation
PCAP	Processor Configuration Access Port
PCW	Processor Configuration Wizard
PDD	Product Definition Document
PE	Processing Element
PFMEA	Process Failure Mode Effect Analysis
PMHF	Probabilistic Metric for random Hardware Failures
PMU	Platform Management Unit
PL	Programmable Logic
PLL	Phase-Locked Loop
POR	Power-On Reset
PS	Processor System (i.e. LPD + FPD)
PST	Process Safety Time
QM	Quality Managed
QOS	Quality of Service
Quad-SPI	Quad Serial Peripheral Interface
RAM	Random Access Memory
ROM	Read Only Memory
RPU	Real-time Processing Unit
RTC	Real-Time Clock
SECDED	Single Error Correction Double Error Detection
SEU	Single Event Upset
SEM	Soft Error Mitigation
SEooC	Safety Element out of Context
SIL	Safety Integrity Level

SLCR	System Level Control Registers
SM	Safety Mechanism
SMC	Secure Monitor Call
SoC	System On Chip
SPFM	Single Point Fault Metric
SRS	Software Requirements Specification
STL	Software Test Library
SW	Software
SYSMON	System Monitor
TCM	Tightly Coupled Memory
TTC	Triple Timer Counters
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VIV	Vivado Isolation Verifier
WDT	Watch Dog Timer
XMPU	Xilinx Memory Protection Unit
XPPU	Xilinx Peripheral Protection Unit

Definitions

This section identifies all definitions used in this safety manual:

- **System/Item**: The final HW ECU (Electronic Control Unit) or ECU set into which the Zynq UltraScale+ MPSoC device has been integrated.
- **User / End User**: The final user of the Zynq UltraScale+ MPSoC in charge of integrating the device into a real application.
- **Application/Application Software**: The software run on the Zynq UltraScale+ MPSoC and other programmable elements in charge of the actual implementation of the safety function.
- **Safety Application**: The subset of the Application Software dedicated to meeting the Item's safety goal. The Safety Application is normally distributed across the system to exploit independence and to facilitate cross-checking of modules. The Safety Application might contribute directly towards the safety function but it will perform tests to verify the operational fitness of Zynq UltraScale+ MPSoC and other elements forming the Item on which the Application Software is reliant.

Faults and Failure Handling Procedures

Origin of Faults

A typical technical system's life cycle consists of three phases: *design*, *production*, and *operation and maintenance*, as shown in Figure 2-1.

Note: Figure 2-1 refers to a product life cycle, not the safety life cycle, which consists of 16 phases, as outlined in IEC61508-1.

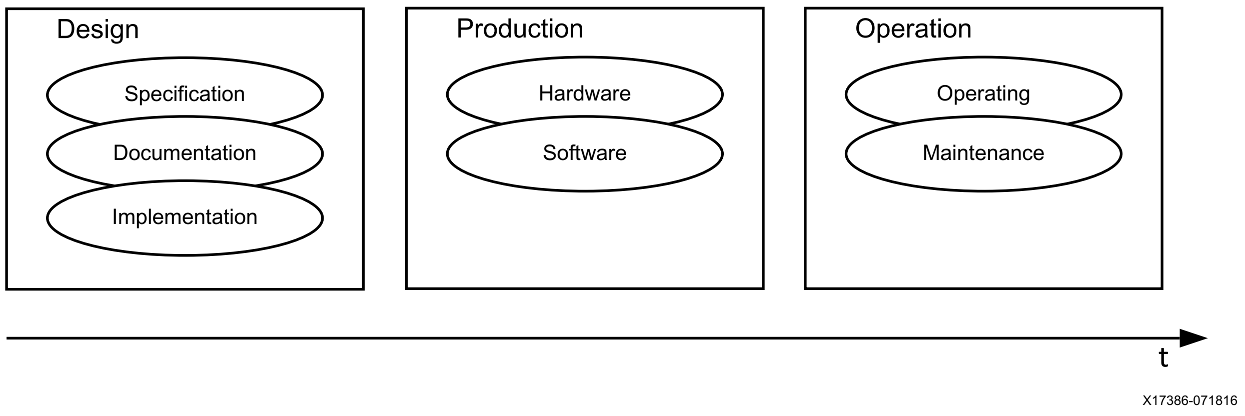

Figure 2-1: **System Life Cycle Phases**

A fault can occur during any of these life cycle phases:

- *Design*: Comprises specification, implementation, and documentation.
    - Specification (such as incorrect definition of operating conditions), implementation (such as software errors which occur regardless of correct specifications), and documentation (which can be misleading and thus cause incorrect usage) are subject to faults.
- *Production*: Includes both manufacturing of hardware and compilation of software.
    - Faults that occur during the manufacturing of hardware and installment of software must also be taken into account.

- *Operation and Maintenance*: Includes both operating and maintenance aspects.
    - In the operation phase, a further distinction between faults caused by disturbances (such as mechanical or electrical influence), wear out, or physical influence must be considered.

Types of Faults

Table 2-1 provides a list of typical faults that can be encountered in an electronic system and the fault handling responsibility. Where applicable, a summary of Xilinx's fault handling capability is provided.

Table 2-1: **Electronic System Faults and Fault Handling**

Fault	Description	Fault Handling
Specification Error	Errors in the transition of user requirements to a system specification that might lead to the violation of safety goals.	• *At item level (user responsibility)*: ◦ This safety manual assumes that the user of the Zynq UltraScale+ MPSoC has fully specified and processed the safety requirements to fulfill the safety goals in compliance with IEC 61508 or ISO 26262, and any other requirement needed for the specific application. • *At element level (Xilinx responsibility)*: ◦ Addressed through the implementation of a product lifecycle compliant to ISO-26262, tailored for a System Element out of Context
External Faults (combined with an inability for the system to process them)	• System-external faults that impact the system (such as affecting the inputs of the system) from performing the safety function. These conditions do not necessarily lead to permanent hardware faults. For example, dust can lead to misreading of an input. These faults are manifested as random or common cause failures (CCF). • In addition, it is important to distinguish between faults which shall be tolerated according to the system specification and those faults which the system is, again according to its specification, not required to process	• *At item level (user responsibility)*: ◦ It is also the user's responsibility to account for the proper CCF and FMEA analysis to mitigate the impact of such failures. Redundancy and on-chip redundancy can improve the reachability of the safety goals. ◦ For faults that the system is not designed to process, the user must bring such inabilities down to the required safety integrity level and provide all of the measures necessary to preserve the integrity level for the whole life time or mission time. ◦ The user shall protect the device from damaging environmental conditions that it be exposed to over the life or the product. For example, encapsulation. • *At element level (Xilinx responsibility)*: ◦ Zynq UltraScale+ MPSoC product characterization according to Automotive AEC-Q003 Guidelines for Characterizing the Electrical Performance of Integrated Circuit Products

Table 2-1: **Electronic System Faults and Fault Handling *(Cont'd)***

Fault	Description	Fault Handling
Implementation Faults	Human-made hardware/software faults that occur in the development of the system and the hardware and software components that make up that system. Included are faults in development tools (e.g. code generators and compilers)	• *At item level (user responsibility)*: ◦ Mitigated by following the development prescriptions defined in IEC 61508 and ISO 26262, best practices, quality control and any other reasonable practice to reach the desired safety integrity level. • *At element level (Xilinx responsibility)*: ◦ Mitigated by following the development prescriptions defined in ISO 26262 tailored for Safety Element out of Context, best practices, quality control and any other reasonable practice to reach the desired safety integrity level. Chapter 3 describes the xilinx development process.
HW Manufacturing Faults	Production faults can be of systematic or random nature (e.g., low level silicon imperfections), with prevalence of the systematic ones, and generally corrected with specific manufacturing process improvement.	• *At item level (user responsibility)*: ◦ Mitigated by following the development prescriptions defined in IEC 61508 and ISO 26262, best practices, quality control and any other reasonable practice to reach the desired safety integrity level. It is the responsibility of the Zynq UltraScale+ MPSoC user to bring such inabilities down to the required safety integrity level and provide all of the measures necessary to preserve the integrity level for the whole life time or mission time. • *At element level (Xilinx responsibility)*: ◦ Mitigated by following the development prescriptions defined in ISO 26262 tailored for Safety Element out of Context, best practices, quality control and any other reasonable practice to reach the desired safety integrity level. Chapter 3 describes the xilinx development process.
HW Faults - Permanent	Caused by wear-out and/or environmental stress (e.g. temperature, voltage, etc.) Operation of hardware outside the specified limits can manifest faults with time. The initial cause might be attributed to different phases. However, the resulting breakdown occurs during the *operation* phase.	• *At item level (user responsibility)*: ◦ It is the responsibility of the Zynq UltraScale+ MPSoC user to operate the device according to its specified limits. Xilinx provides a *Reliability Report and Qualification Data Report* (UG116) to assist the customer in designing the product to maximize the reliability, availability, and lifetime. • *At element level (Xilinx responsibility)*: ◦ Zynq UltraScale+ MPSoC probability metrics of hardware failure are presented in the Zynq UltraScale+ MPSoC FMEDA Report. The SPFM and LFM For the low power domain sub-element are also reported.

Table 2-1: **Electronic System Faults and Fault Handling *(Cont'd)***

Fault	Description	Fault Handling
HW Faults - Transient	Faults caused by neutrons, alpha particles, and other types of radiation. These errors are also known as *soft errors* and manifests themselves as random errors. Their rate depends mainly on the altitude of operation (higher altitude translates to a higher likelihood of error) and on the impurities within the device packaging materials. No semiconductor is totally immune to such radiations.	• *At item level (user responsibility)*: ∘ Xilinx offers a suite of design tools, upset rate estimators, hardware and software IP, upset detection and upset correction solutions. Appropriate selection and application of these techniques and circuits will minimize the effects of upsets, and in some cases, virtually eliminate all possibility of faults arising from soft upsets from occurring at all. It is the responsibility of the Zynq UltraScale+ MPSoC user to follow Xilinx recommendations when using the design rules and mitigation techniques provided by Xilinx. This Safety Manual makes recommendations that enable the user to achieve soft error mitigations that fully ensure the highest safety integrity level when using the Processor System (PS). When also using the Programmable Logic (PL) the user should consider the application of hardware design techniques such as Triple Modular Redundancy (TMR) and the Isolation Design Flow (IDF) as well as using the Soft Error Mitigation (SEM) IP core. • *At element level (Xilinx responsibility)*: ∘ Zynq UltraScale+ MPSoC probability metrics of hardware failure are presented in the Zynq UltraScale+ MPSoC FMEDA Report. For the low power domain sub-element the SPFM and LFM are also reported.
Malicious Faults	Security threats play an increasing role in safety systems. The hardware and software must be reasonably protected from exploitation.	• *At item level (user responsibility)*: ∘ It is the user's responsibility to enable and apply the security features Xilinx have provided the Zynq UltraScale+ MPSoC with. • *At element level (Xilinx responsibility)*: ∘ The Zynq UltraScale+ MPSoC has been designed specifically to protect against threats to security. A *Root of Trust Secure Boot Process* (XAPP1175), combined with voltage and temperature monitoring, ARM® TrustZone®, Security Monitor IP, and other techniques are available to mitigate the possible loss of safety functions due to security breaches.

Chapter 3

Zynq UltraScale+ MPSoC Development Process - Systematic Fault Management

Hardware Development Process

Xilinx devices can be found in a broad range of applications including Mars probes, robotic surgery systems, wired and wireless networking infrastructure, high definitions video cameras and displays, industrial manufacturing and automation systems, and automotive. The Xilinx development process is a phased development flow designed to meet the most stringent requirements of each of the markets served, while providing the infrastructure required to control systematic faults.

The Zynq UltraScale+ MPSoC Development Process is a Quality Managed (QM) process. The Xilinx Quality Management System is a mature system that has a long history of third-party certification. Xilinx is currently certified to the TL9000 Quality Management System since 2004 and has held ISO9001 certification prior to 2004. Xilinx supply chain requirements include TS16969 / ISO9001 certification. These certification requirements ensure that all processes throughout the entire product lifecycle are subject to annual third-party certification audits confirming compliance to Xilinx internal requirements as well as the requirements of the ISO9001 / TS16949 / TL9000 standards. The Xilinx Quality Manual (QAP0002) as well as quality certification information is available online at: http://www.xilinx.com/support/quality.html.

Absolute quality defines Xilinx's commitment to excellence by developing control plans, DFMEA and manufacturing PFMEAs. Stringent control processes and maverick controls are focused to prevent quality issues at all stages of manufacturing. Xilinx eliminates significant sources of variability to deliver the highest levels of product quality and reliability by using innovative statistical techniques and near real-time controls. Xilinx delivers continual improvement through the use of comprehensive root cause analysis to resolve and prevent problems. Together with Xilinx suppliers, Xilinx leverages a team-based, Eight Discipline (8D) investigative approach using a data driven verification methodology to develop permanent and effective solutions. As stated previously, Xilinx regularly plans and executes audits of all aspects of the management systems to validate and verify effectiveness and adherence to defined procedures and standards. Management is dedicated to the timely correction of deficiencies and demonstration of a culture of continual improvement.

Hardware Development and Production Process

The Xilinx product life cycle model includes product definition, design and development, pre-production qualification, ongoing manufacture and delivery, and obsolescence. Approval of senior management is required to move on at the completion of each phase. The Xilinx product life cycle phases are depicted in Figure 3-1.

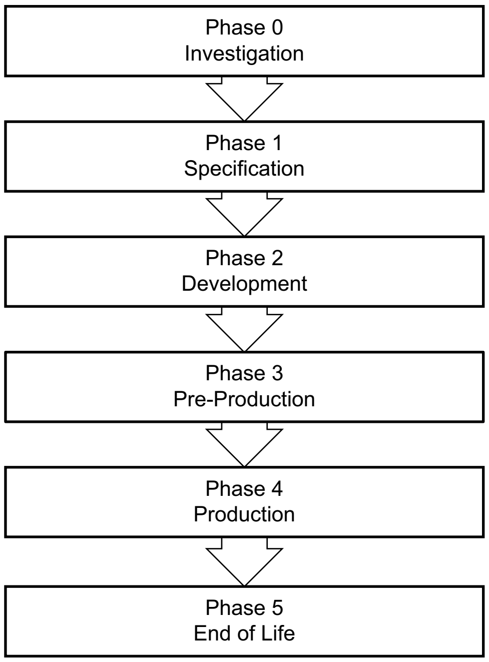

Figure 3-1: **Xilinx Product Development Process**

Product definition begins in Phase 0 with the product concept. Specific customer requirements are gathered and the definition of controls and test needed for compliance are captured. The output is a *product concept*.

Phase 1, Specification, further defines the detailed product requirements. Product Definition Documents (PDD) and other architectural specifications are generated. In addition, a Verification Plan, Development Schedule, and Resource plan are approved.

Phase 2, Development, is where the actual product design files are created and validated prior to committing the design to hardware. Holistic simulation checks functionality of the design against the software tools description of the device. This phase culminates in tape-out of the first device in the family.

Phase 3, Pre-production, is where the design is validated in hardware. Characterization data is collected and analyzed. Qualification tests are completed. Xilinx uses stringent testing and verification to mitigate risk of errors in its development process and as result in the user's delivered development tools. Early testing, real-time feedback, metrics tracking, best practices and other state of the art error minimization, defect control, detection and removal, is used. When it is determined that the product meets the product requirements it will move into Phase 4, Production.

Phase 4, Production, is where the data sheet and other specifications are final. Management of the product moves to the Product Test, Manufacturing Operations, and Technical Support teams.

Phase 5, End of Life defines the process of removing the product from active production through Product Change and Discontinuation Notices.

Note that the development process described is specific to the silicon manufacture. The software development process, and resulting certification of the Xilinx Development Tools is documented in the *Vivado Safety Manual* (UG1187).

Software Development Process

The progress and quality of the software release is tracked during development through intermediate development milestones. The SW development lifecycle phases are depicted in Figure 3-1.

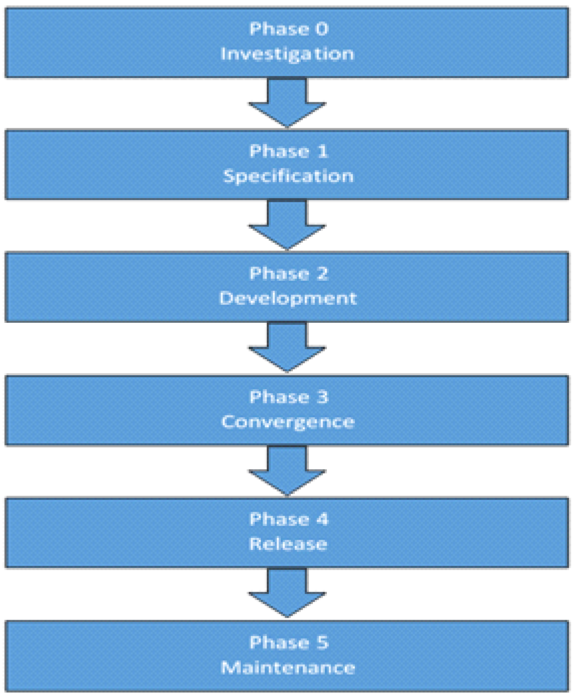

Figure 3-2: **Software Product Life Cycle**

Phase 0, Investigation, involves the requirements gathering based on the market analysis and the product initiatives. The output is the Market Requirements Document (MRD).

Phase 1, Specification, is the phase where both the Marketing and Development teams agree on the marketing requirements.

Phase 2, Development, is the phase in which the Development team commit to realize the specification. This involves various sub phases: planning of resources and schedules, software requirements specification (SRS), design which comprises architecture and detailed design, implementation, verification, and validation. At this stage, all of the committed features are marked to be complete, and checked in to revision control.

Phase 3, Convergence, is the phase which allows incorporation of changes resulting from quality tuning or late feature implementation. The changes to the revision control is through change management with appropriate approvals from designated persons.

Phase 4, Release, is the actual release of source code to the end customer

Phase 5, Maintenance, involves addressing of any field issues through change management process.

Safety Element Out of Context Approach

The Zynq UltraScale+ MPSoC is not an *item* in the sense of ISO 26262, but is an *element* (developed by Xilinx) of an *item* developed by an end user, that is, a Xilinx customer at a later point in time. Due to this reason, the Zynq UltraScale+ MPSoC product has followed a *Safety Element out of Context* (SEooC) development cycle through the proper tailoring of the ISO-26262 standard, performed according to Clause 9 of ISO-26262 Part 10. The validity of such a tailoring has been checked by an independent Functional Safety Assessor (EXIDA).

According to this SEooC approach, the Zynq UltraScale+ MPSoC product has been developed to satisfy a collection of functional and safety requirements which are assumed to be generated by the items that will integrate the Zynq UltraScale+ MPSoC (see Figure 3-3).

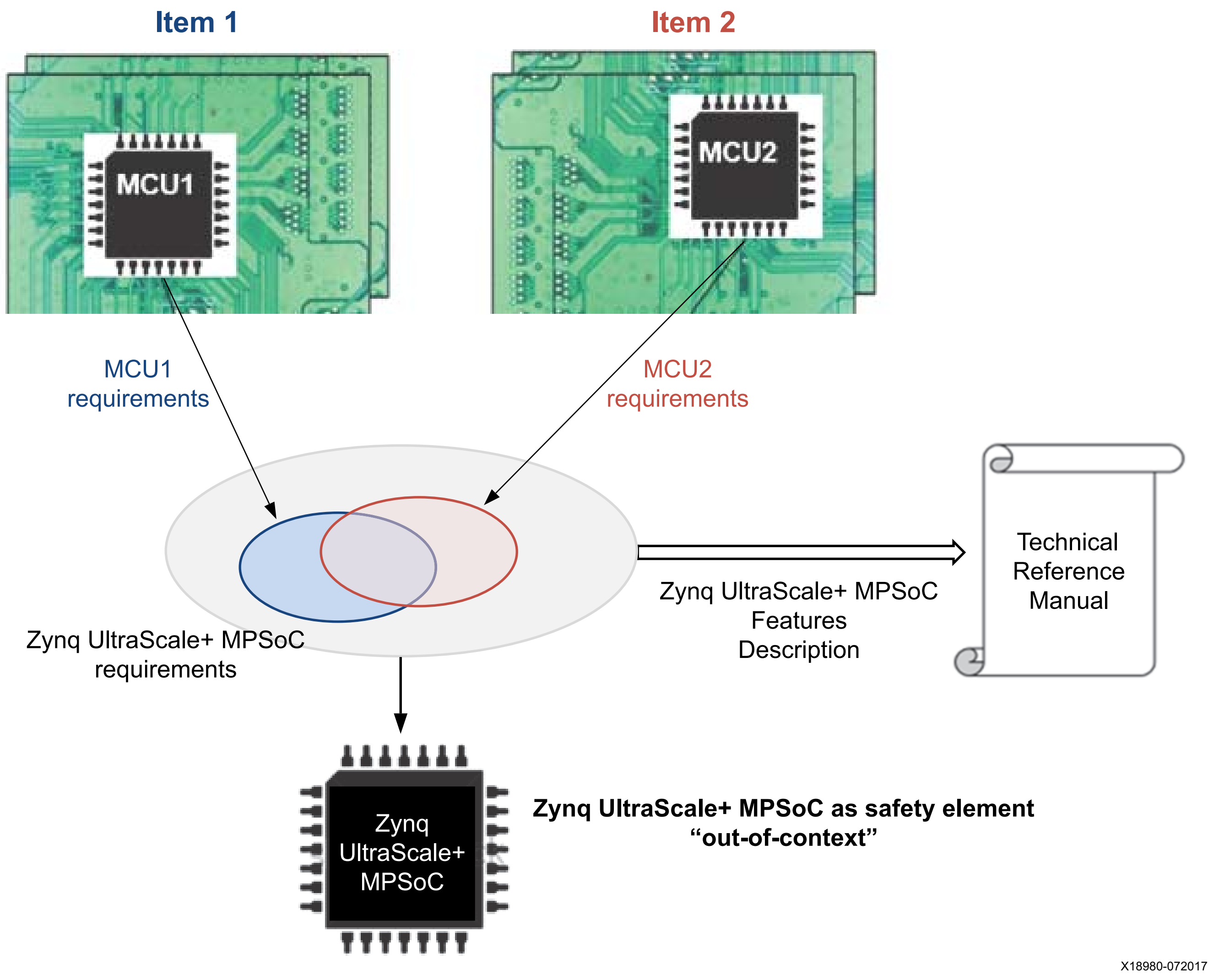
Item 1
Item 2
MCU1
MCU2
MCU1 requirements
MCU2 requirements
Zynq UltraScale+ MPSoC requirements
Zynq UltraScale+ MPSoC Features Description
Technical Reference Manual
Zynq UltraScale+ MPSoC
Zynq UltraScale+ MPSoC as safety element "out-of-context"
X18980-072017

Figure 3-3: **Safety Element Out of Context Approach**

Assumptions of Use Generated from Safety Concept and Safety Analyses

The Safety Concept makes certain assumptions about how the Zynq UltraScale+ MPSoC will be used, the software which will be implemented and run on the Zynq UltraScale+ MPSoC when integrated into an item, and the other components of the Item. Furthermore, the Safety Analyses performed based on an Zynq UltraScale+ MPSoC detailed designs also makes assumptions on Zynq UltraScale+ MPSoC usage, software, and external components. Those assumptions are called Assumptions of Use (AoU), see Figure 3-4.

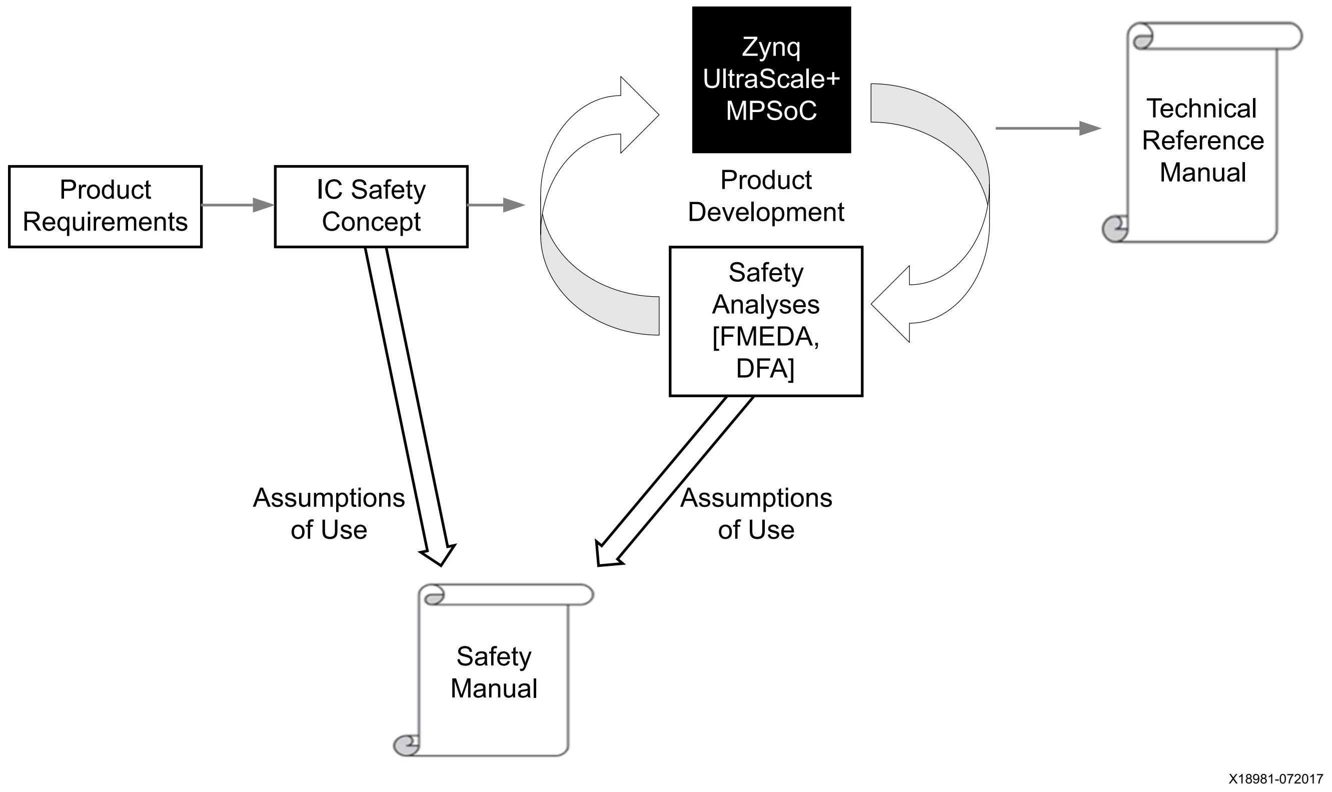

Figure 3-4: **Generation of Assumptions of Use**

These assumptions are documented and provided to the item integrator through this Safety Manual (see Chapter 5). These assumptions are meant as requirements for any later software development and system integration. Failure Mode Effect and Diagnostic Analysis (FMEDA) confirms that the Zynq UltraScale+ MPSoC device supports safety levels up to and including SIL3 and ASIL-C when all assumptions of use have been followed (see Table 4-1, page 22).

Along with the Zynq UltraScale+ MPSoC hardware element, the product description documentation (Technical Reference Manual, Data Sheet, Errata Sheet) and safety related documentation (Safety Manual) are being provided to parties (i.e. Item Integrators) which integrate the Zynq UltraScale+ MPSoC into a safety related Item.

Zynq UltraScale+ MPSoC Safety Concept

Architecture Description

As shown in Figure 4-1, the Zynq UltraScale+ MPSoC can be divided in three main sub-systems:

- Low Power Domain (LPD), based on two ARM Cortex-R5 processors
- Full Power Domain (FPD), based on four ARM Cortex-A53 processors
- Programmable Logic (PL), based on Xilinx programmable technology.

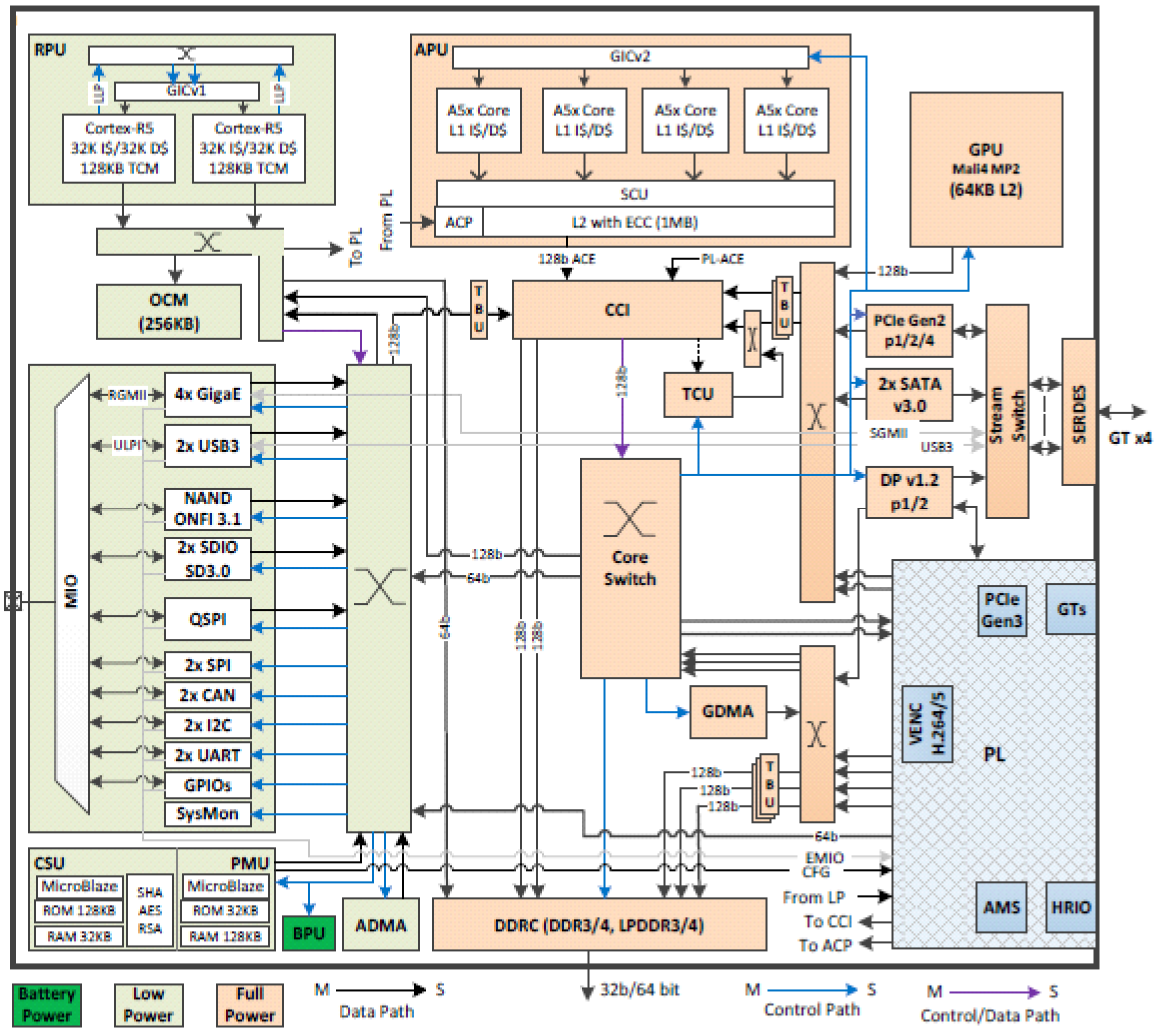

Figure 4-1: **Zynq UltraScale+ MPSoC Architecture Diagram and its Power Domains**

For functional safety related applications, the Zynq UltraScale+ MPSoC is defined (see Figure 4-2) as an SEooC playing the role of a processing element (or part of it). The Zynq UltraScale+ MPSoC is supposed to be connected to remote controller(s) by means of a communication bus. The communication bus is directly or indirectly connected to sensor(s) and actuator(s).

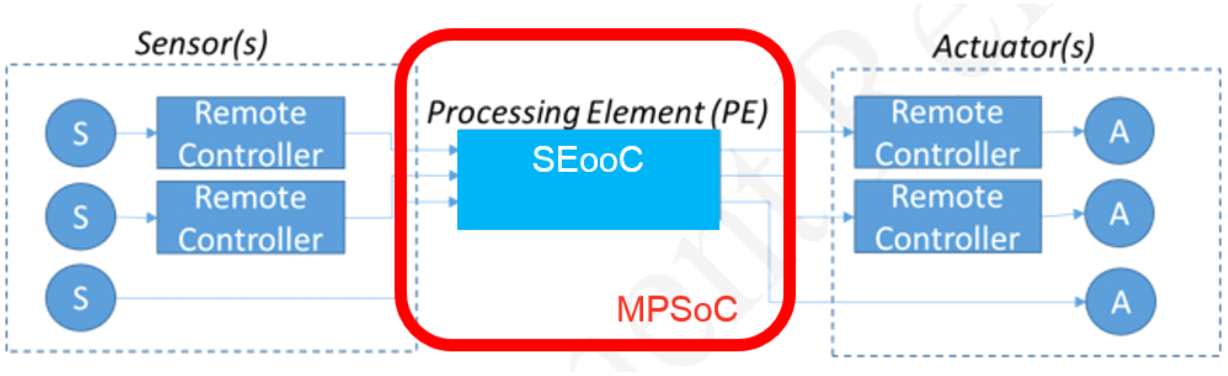

Figure 4-2: **Zynq UltraScale+ MPSoC as Safety Element Out of Context**

Each of the three subsystems/power domains of the Zynq UltraScale+ MPSoC might have a different role in the SEooC. For example, in the case of an automotive ADAS application (e.g. autonomous emergency braking), consider the architecture shown in Figure 4-3.

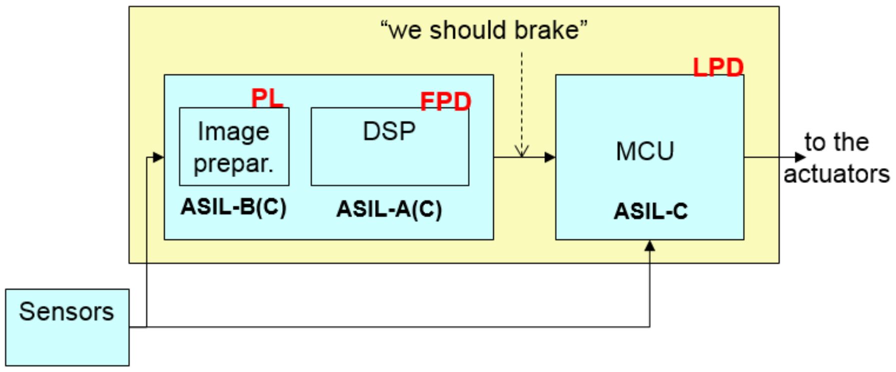

Figure 4-3: **Example of an ADAS-AEB Application Using the Zynq UltraScale+ MPSoC**

In this architecture:

- The image preparation (that might be a sensor-dependent or end user-dependent function) could be implemented in the PL
- The DSP (that needs a very high processing power) could be implemented in FPD
- The MCU (that needs a very high level of safety because it controls the actuators) could be implemented in the LPD.

In an industrial application with a very high safety critical I/O control function, consider the architecture shown in Figure 4-4.

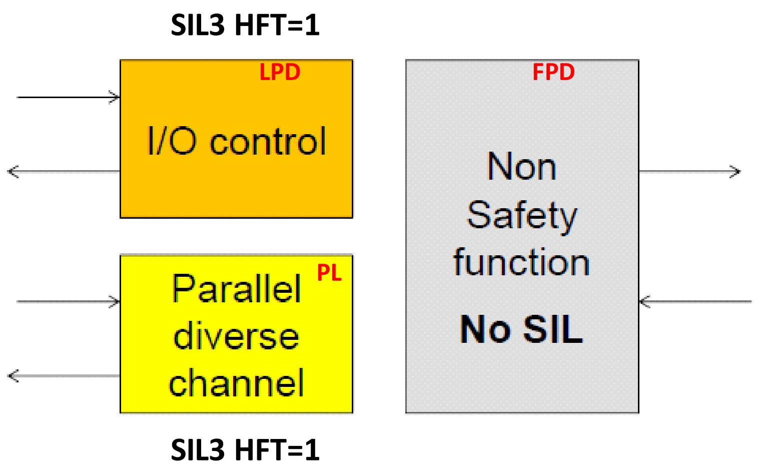

Figure 4-4: **Example of an Industrial Application Using the Zynq UltraScale+ MPSoC**

In this architecture:

- The safety-related I/O control could be implemented in the LPD
- The parallel diverse channel could be implemented in the PL
- The non-safety function could be implemented in the FPD

Possible Sub-system Configurations for Safety Related Applications

The Zynq UltraScale+ MPSoC can potentially support the safety configurations listed in. Table 4-1.

Table 4-1: **Safety Configurations**

System Configuration	LPD (RPU) @HFT=0	FPD (APU) @HFT=0	PL @HFT=0	No. of Safety Channels	HFT	Notes
1	SIL3/ASIL-C	Non Safety	Non Safety	1	0	-
2	SIL3/ASIL-C	Non Safety	SIL2/ASILB(1)	2	1 (LPD+PL)	Achievable if a certain level of isolation/separation is guaranteed between the LPD and PL. Moreover, interferences of FPD to LPD or PL shall be avoided or detected (see XPPU Configuration, page 43).

Table 4-1: **Safety Configurations *(Cont'd)***

System Configuration	LPD (RPU) @HFT=0	FPD (APU) @HFT=0	PL @HFT=0	No. of Safety Channels	HFT	Notes
3	SIL3/ASIL-C	SIL2/ASILB(1)	SIL2/ASILB(1)	2+1	1 (LPD+PL or FPD+PL)	Achievable if a certain level of isolation/separation has to be guaranteed between LPD, FPD and PL (see XPPU Configuration, page 43).

Notes:

1. Listed here as example case, but are not PL safety design requirements. PL@HFT=1 adds to PS requirement for independence of channels.

For the assumptions of use currently presented in this Safety Concept document, the following figures describe the possible combinations of use of Zynq UltraScale+ MPSoC partitions and the respective level of compliance:

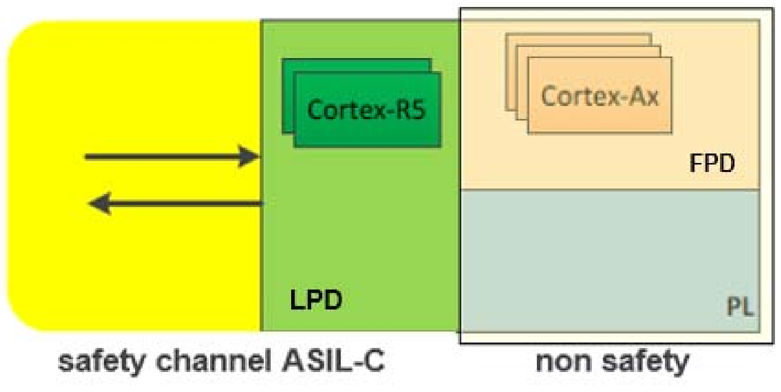

Figure 4-5: **Zynq UltraScale+ MPSoC Partitions - Configuration 1**

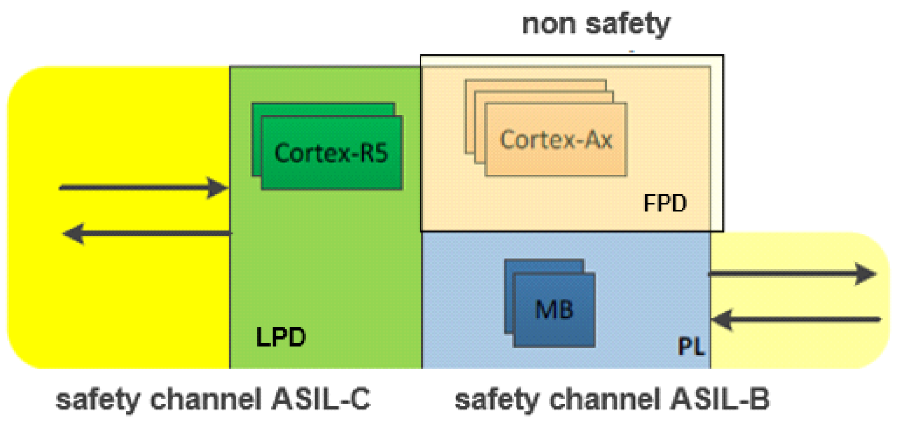

Figure 4-6: **Zynq UltraScale+ MPSoC Partitions - Configuration 2**

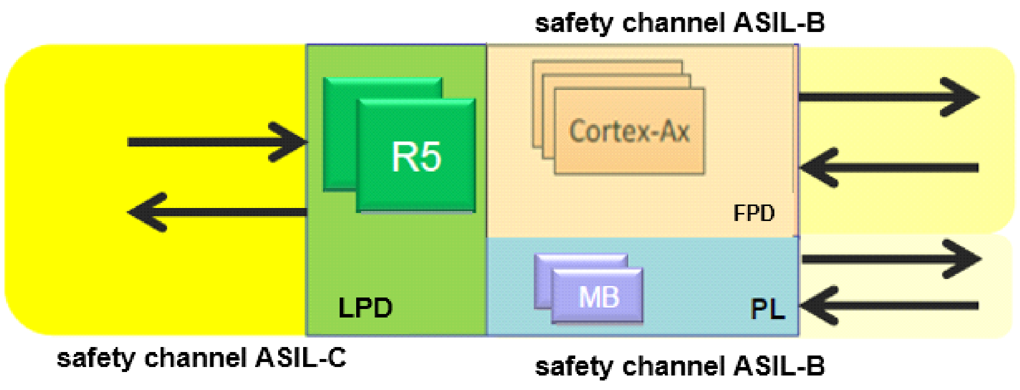

Figure 4-7: **Zynq UltraScale+ MPSoC Partitions - Configuration 3**

Assumed Zynq UltraScale+ MPSoC SEooC Safety Goal and General Assumptions of Use

This section defines the safety goal of the Zynq UltraScale+ MPSoC and the high level assumptions of use.

Safety Goal

A failure in the LPD subsystem caused by a hardware fault shall not cause the LPD to go in an unsafe state for a time greater than FTTI MRD_SG001.

> ***Note:*** The tags with the format MRD_xxx, SM_xxx and SC_xxx are used by Xilinx for requirement traceability and should be ingnored by the reader.

Unsafe state is any state not listed as a *safe state* in Operating States – Safe States, page 25

This means that for the scope of this safety concept we are targeting Configuration 1 only (Figure 4-5).

If an item integrator using the Zynq UltraScale+ MPSoC wants to implement Configuration 2 or 3, the safety concept shall be extended to the PL partition or both the PL and FPD partitions MRD_AOU001.

> ***Note:*** If a Xilinx customer wants to take one of those two options, Xilinx will support the extension of the safety concept, but this will be done outside the scope of this document.

ASIL level

The LPD subsystem of the Zynq UltraScale+ MPSoC shall be considered as an ASIL-C Safety Element out of Context. This means that an Automotive Application developed according to

ISO-26262 ASIL-C can consider the LPD hardware and the software running on the LPD released by Xilinx as a SEooC.

Operating States – Safe States

MRD_AOU024, MRD_AOU025, MRD_AOU057

The safe and non-safe operating states of the Zynq UltraScale+ MPSoC are identified in Figure 4-8.

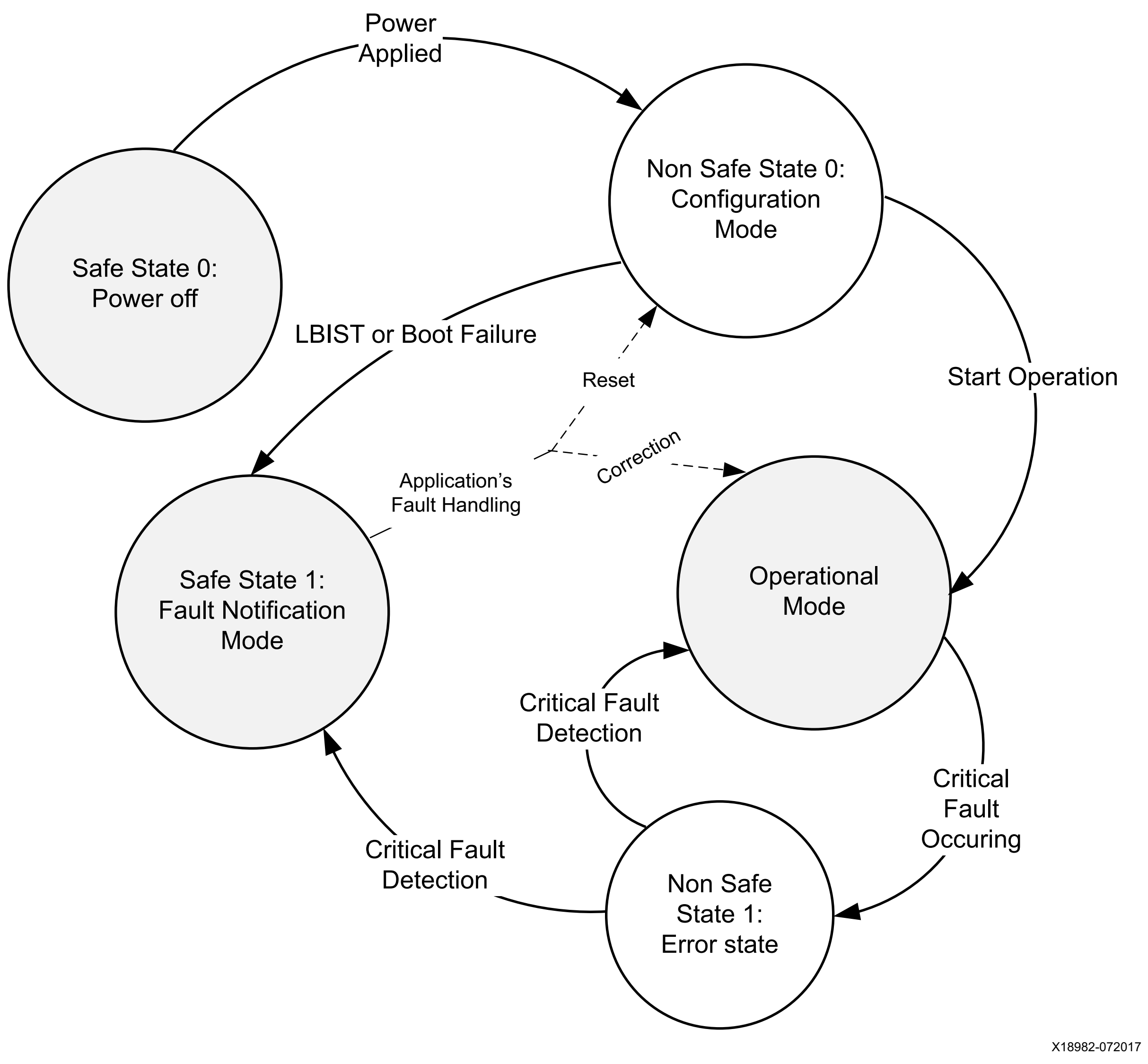

Figure 4-8: **Zynq UltraScale+ MPSoC Operating Safe and Non Safe States**

Notes related to Figure 4-8:

- Safe state 0: The device is de-energized.

- Non Safe state_0 (Configuration Mode): The device has been energized and it is in the start-up phase.
    - This state shall be considered a non-safe state. While the user application software is configuring the device, performing diagnostic checks, calibration and system setup, it is assumed that the Zynq UltraScale+ MPSoC device is unable to perform safety-related functions. From the Item's perspective this state would generally take place at a time when the safety-related function is not required (e.g. when a car is stationary during the key-on phase).
- Operational Mode: The device is working according to the functional specifications stated in the Zynq UltraScale+ MPSoC data sheet (DS925) [Ref 5], and the PS_ERROR_OUT signal is Low.
    - The transition from Configuration Mode to Operational Mode shall be handled by the user application, which is managing the item configuration and has the visibility of when the configuration is complete and when operations can start.
- Non Safe state 1 (Error Mode): If a critical fault occurs the LPD goes in an undefined state until the fault is detected. This is the error state which is going to be prevented or quickly skipped thanks to the safety countermeasures.
- Safe state 1 (Fault Notification Mode): The PMU error handling and propagation logic circuit is informed about the presence of a device failure and the presence of a fault is indicated to the outside world.
    - Failures detected by the Logic Built-In Self Test (LBIST) or fatal errors detected in any of the circuits used during the configuration process result in assertion of the PS_ERROR_OUT pin, PS_ERROR_STATUS pin or both, and the device will not boot.
    - The method of reporting failures detected during operation are defined by the user application. Notably, the enables and masks associated with propagating the error status bits contained in the ERROR_STATUS_1 and ERROR_STATUS_2 registers will define the generation of interrupts, resets or assertion of the PS_ERROR_OUT pin as required for each source of error.
    - The default settings of the masks associated with error reporting during the configuration mode are shown in Table 4-2, page 34 (List of Errors in the Error Handling Logic and the Reset State of their Masks). Any adjustments to the masks by the user application must be made before transition to the operational mode.

Fault Tolerant Time Interval (FTTI) – Process Safety Time (PST)

MRD_AOU002

Because the Zynq UltraScale+ MPSoC has been developed as a *safety element out of context*, fault handling procedures vary from system to system. However, the time in which a system must detect and respond to a failure must be specified.

The single-point fault tolerant time interval (FTTI) (i.e. ISO 26262), or equivalent process safety time (PST) (i.e. IEC 61508) is the time span during which a fault can be present in an item before a hazardous event occurs without a safety mechanism being activated (see Figure 4-9 and Figure 4-10). With regard to the safety application implemented within the LPD of the Zynq UltraScale+ MPSoC it is assumed that the item's FTTI is equal to or greater than 100 ms. Meeting the item’s FTTI requirement is the responsibility of the user when integrating the software test libraries and implementing the assumptions of use.

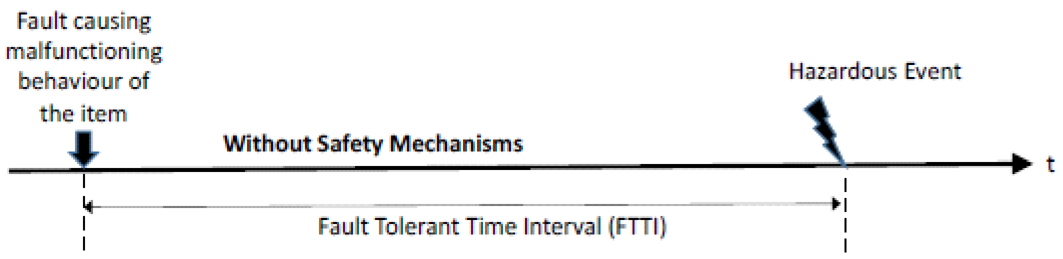
Fault causing malfunctioning behaviour of the item
Hazardous Event
Without Safety Mechanisms
t
Fault Tolerant Time Interval (FTTI)

Figure 4-9: **Fault Tolerant Time Interval**

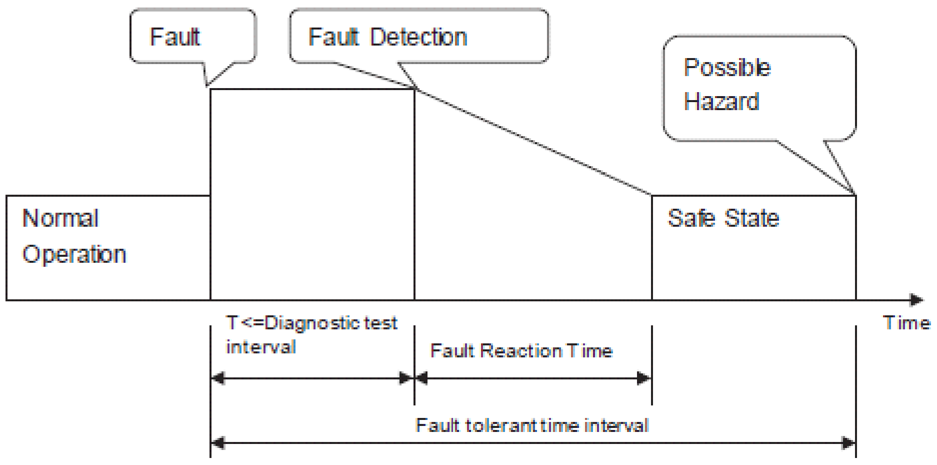

Figure 4-10: **Fault Reaction Time and Fault Tolerant Time Interval**

Latent Faults and Multiple Point Faults Detection Interval

As per ISO-26262 definition, latent faults are multiple-point faults whose presence is not detected by a safety mechanism nor perceived by the driver/user within the multi-point fault detection interval. This latter is the time span to detect multiple-point faults before they can contribute significantly to the probability of the combination of several independent faults leading to the violation of the safety goal. The multi-point fault detection interval time shall be ≤10 hours.

Note: This value is considered adequate for an Automotive application. Based on the current estimation, a value of MPFDI ≤10 will have a negligible impact on the residual FIT rate (i.e., the PMHF metric). In case the application needs a greater MPFDI value (e.g., for trucks or industrial applications), the impact on the PMHF will be revisited in a future revision of this document.

Because the detection of multi-point faults is performed during system start-up through the LBIST, MBIST, and fault injection, the operational time shall not be more than 10 consecutive hours.

Overview of The Safety Mechanisms Implemented in the Zynq UltraScale+ MPSoC

This section gives an overview of the safety mechanisms implemented in the Zynq UltraScale+ MPSoC. LPD detailed architecture and safety mechanisms are shown in Figure 4-11.

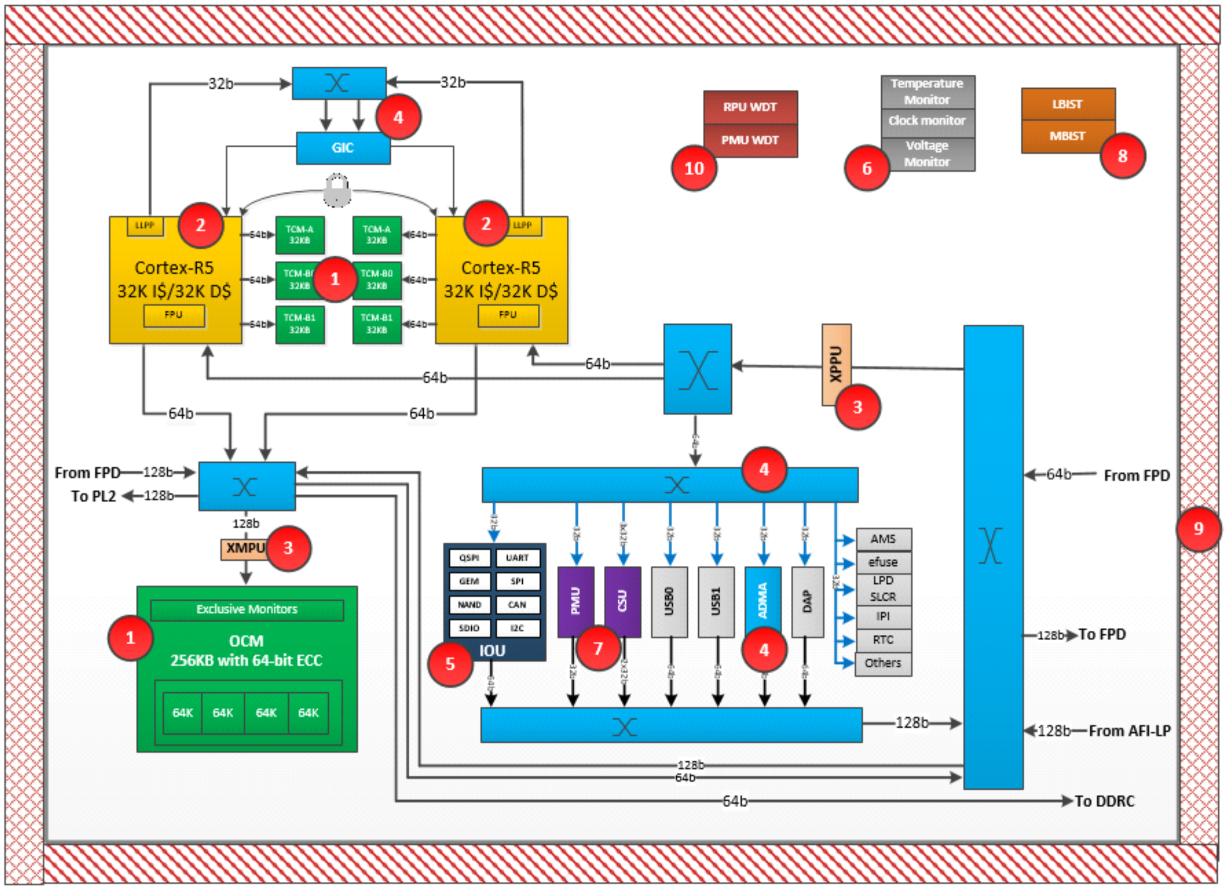

Figure 4-11: **LPD Detailed Architecture and Safety Mechanisms**

Single Point Fault Detection Measures

- ECC covers OCM, PMU-RAM, CSU-RAM, and RPU L1 cache and TCM memories (see "1" in Figure 4-11).
  - ECC RAMs address decode error detection.
    - Separate RAMs for ECC syndrome and data
  - 4:1 or greater interleaving of memory cells protected by ECC. Each ECC facilitates the detection and correction of a single bit error within the data word that it

protects. Interleaving assigns physically adjacent memory cells to different words minimizing the potential for uncorrectable errors to occur even if multiple bit upset of physically adjacent memory cells should occur. For example, a double bit upset of physically adjacent memory cells will result in two different words each containing a single bit error. Both of these errors will be detected and corrected using the respective ECC for each word.

- Hash validation of CSU BootROM contents at every boot (see "7" in Figure 4-11)
- Lockstep and redundancy covers R5 (see "2" in Figure 4-11)
  - R5 lockstep with physical and temporal diversity
  - Redundant logic in critical control logic such as R5 lockstep checkers
- PMU and CSU implemented with redundancy (see "7" in Figure 4-11)
  - TMR (triple module redundant) processor cores with physical diversity
  - Triple redundant flip-flops for critical control bits such as security state
- XMPU and XPPU protect memory space (see "3" in Figure 4-11)
- Watch-dog timers
  - Watchdog timers provided in LPD and FPD
  - LPD watchdog timers for RPU and PMU (see "10" in Figure 4-11)
- Software Test Library
  - See *Zynq UltraScale+ MPSoC Software Safety User Guide* (UG1220) for details
- End-to-end software protocol (see "4" and "5" in Figure 4-11)
  - AXI interconnect, IOU, and ADMA operational coverage

Common Cause Failure Measures

- System Monitoring (see "6" in Figure 4-11)
  - Voltages monitoring
  - Temperature monitoring
  - Clock frequency monitoring
- Error Management (see "7" in Figure 4-11)
  - Error management is handled and implement within the PMU
  - Errors are signaled as interrupts and copied to the PL
  - All hardware and software errors captured in ERROR_STATUS_1 and ERROR_STATUS_2 registers are visible to the PMU, RPU, APU, and PL
- Monitoring of activation of CCF (common cause failures) by the PMU
  - MBIST, LBIST, SCAN, reset, power control (see "7" and "8" in Figure 4-11)

- Hang protection
  - Cleanup of outstanding transactions under partial reset
- Meta-stability errors
  - PS uses redundant flip-fops in selected clock crossings
- Aging Errors
  - Large on-chip variation (OCV) margin accounts for aging effects
- Mechanical Errors
  - Package quality is highly tested

Latent Fault Measures

- All check logic such as XMPU, lockstep and ECC checkers are checked at boot by LBIST (see "8" in Figure 4-11)
- All LPD memories can be tested at full processing speed during boot by MBIST
  - Most memories (excluding PMU and CSU program RAMs) can be tested on demand during execution
- The functionality and status of TCM, OCM, PMU RAM, R5 Lockstep, PMU, XMPU, XPPU, clocks, voltages, and temperatures can be tested and evaluated periodically throughout operational mode.
  - See *Zynq UltraScale+ MPSoC Software Safety User Guide* (UG1220) for details

Isolation Measures

(See "9" in Figure 4-11.)

- LPD supports isolation from the rest of the system
- Flexible reset management
  - Enables use of processors for redundant processing
  - Reset management is implemented in the PMU
  - Independent reset for LPD, FPD, PL, and PS-only
- Independent power domains
  - LPD, FPD, and PL
  - Built-in AXI timeout on PL master interfaces

Additional Measures

Safety features for the non-LPD Zynq UltraScale+ MPSoC components are additional measures that can be used for Configuration 2 and 3 shown in Figure 4-6 and Figure 4-7, therefore outside the LPD scope.

- SEU Mitigation for external memory
  - DDR interface supports ECC for 32-bit and 64-bit words
    - Double error detection
    - Single error correction
  - ECC support for APU L2, L1-D memories
  - Parity support for APU L1-I memories
  - QOS management to prevent live-lock
    - QOS controls on masters
    - QOS management in PS AXI
    - QOS management in PS DDR controller
- PL or Multi APU cores can provide redundant processing
- All FPD memories can be tested at full processing speed during boot by MBIST
- Leverage of PL for implementation of safety features
  - Provides HFT channel capability
  - Provides error logging
  - PL can remain active if PS is reset due to error

Safety Measures Implemented in Software

Some error checking is required to be implemented in software. See *Zynq UltraScale+ MPSoC Software Safety User Guide* (UG1220) for more information.

Fault Handling Procedures

When a safety mechanism detects a fault, the corresponding error state must be indicated. The Zynq UltraScale+ MPSoC architecture provides aggregation of error indication from internal safety mechanisms using the platform management unit (PMU) which includes peripheral logic dedicated to the handling of errors.

The PMU is responsible for capturing all errors within the device, reporting these errors to the outside world, and taking the appropriate action with respect to each error. The PMU includes the necessary registers, logic, and interfaces for handling these functionalities (see Figure 4-12).

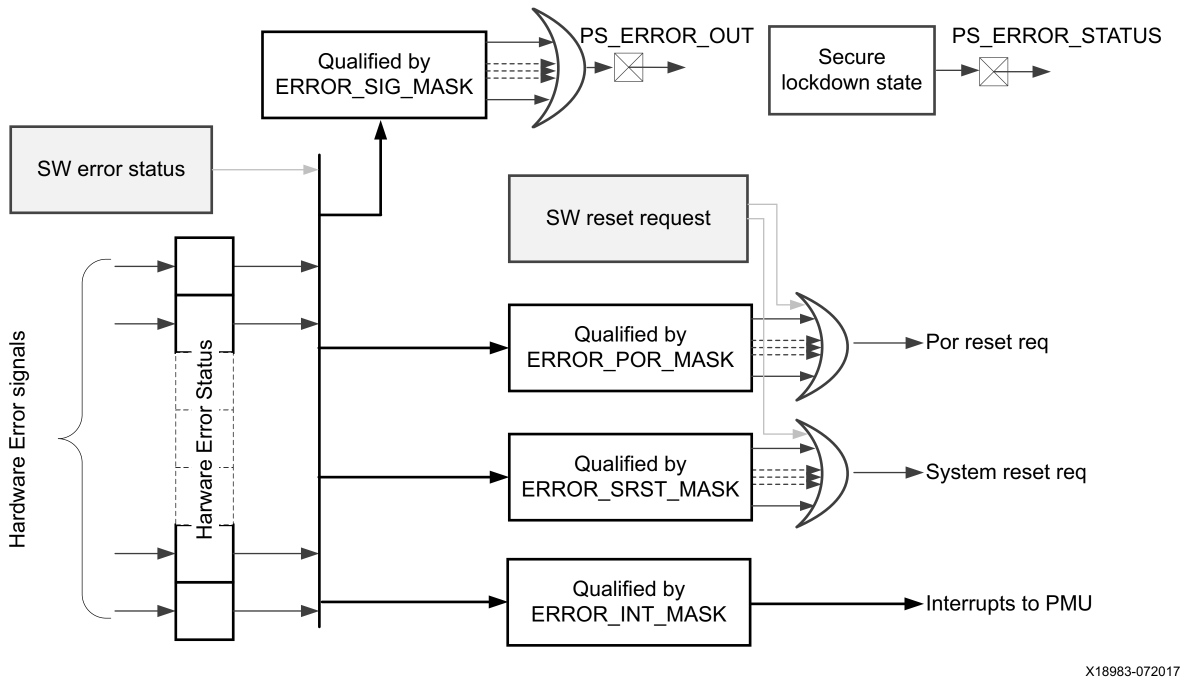

Figure 4-12: **Dedicated Error Handling Logic Within the PMU**

The PMU provides a collection of error input signals that route all system-level hardware errors to the PMU in order to be captured. These errors are recorded in the Error Status registers (ERROR_STATUS_1 and ERROR_STATUS_2) within the PMU and are not cleared even during a system reset or an internal POR. Error status will only be cleared when the external PS_POR_B pin is used or when the PMU explicitly writes a 1 to each corresponding error status bit within the Error Status registers. All errors can generate an interrupt to the PMU. This interrupt can be masked per error.

The propagation of all errors to the Error Status registers can be disabled by using the bits in the Error Enable registers (ERROR_EN_1 and ERROR_EN_2) global registers in the PMU. See *PMU Error Handling and Propagation Logic in Chapter 6* of UG1085 [Ref 4], which includes the description of these registers and their reset states.

The PMU can also capture four software-generated errors. The CSU can record ROM pre-boot errors and the PMU can record pre-boot, service, and firmware errors by setting the appropriate bits in the ERROR_STATUS_2 register. Similar to hardware errors, software-generated errors can be masked per error and are only cleared by using the external PS_POR_B pin or by explicitly writing a 1 to the corresponding bit in the ERROR_STATUS_2 register. All except the PMU pre-boot execution error (PMU_PB) can generate an interrupt to the PMU.

For each of the errors that are processed by the error handling logic, the user can decide what action should be taken when the error occurs. The possible scenarios would be one of a combination of the following choices:

- Assertion of the PS_ERROR_OUT signal on the device
- Generation of an interrupt to the PMU processor
- Generation of a system reset
- Generation of a power-on reset

Table 4-2 shows errors routed to the error propagation logic and the default state of their masks. It should be recognized that Table 4-2 reflects the reset states of the ERROR_SIG_MASK_1/2, ERROR_INT_MASK_1/2, ERROR_SRST_MASK_1/2 and ERROR_POR_MASK_1/2 registers that define the propagation of errors reported to the ERROR_STATUS_1/2 registers. The propagation of errors from their sources to the majority of status bits contained in the ERROR_STATUS_1/2 registers are themselves subject to the reset states of corresponding error-specific masking registers.

The user application shall program the PMU and configure the error handling logic within the PMU according to the Application Safety Concept. Xilinx SM tracking tag: [SC_AOU302]

Table 4-2: **Errors Routed To Error Propagation Logic and Mask Default State**

Error Source	No. of Error Bits	Reset State of Masks (M=Masked, U=Unmasked) PS_ERROR_OUT, PMU Int, SRST, POR	Notes
Software Errors			
PMU ROM Preboot Error	1	U / M / M / M	PMU Preboot Error Flag bit from PMU_PB_ERR register to signify an error during PMU Preboot ROM Execution.
CSU BootROM Error	1	U / M / M / M	CSU BootROM Error Flag bit from CSU_BR_ERR register to signify an error during CSU BootROM Execution. Refer to CSU BootROM Spec for the definition of these bits.
PMU ROM Service Error	1	U / M / M / M	Error Flag bit from PMU_SERV_ERR register to signify an error during execution of a PMU service.
PMU Firmware Error	4	U / M / M / M	Error bits from Firmware field of the PMU_SERV_ERR register to signify errors during Firmware execution.
Hardware Errors			
PMU Uncorrectable HW Error	1	U / M / M / M	Uncorrectable PMU Error. Includes ROM validation, TMR, uncorrectable RAM ECC, and Local Register Address Errors.

Table 4-2: **Errors Routed To Error Propagation Logic and Mask Default State *(Cont'd)***

Error Source	No. of Error Bits	Reset State of Masks (M=Masked, U=Unmasked) PS_ERROR_OUT, PMU Int, SRST, POR	Notes
DDR Uncorrectable ECC Error	1	M / M / M / M	Uncorrectable ECC Error during a DDR access.
OCM Uncorrectable ECC Error	1	M / M / M / M	Uncorrectable ECC Error from during an OCM access.
RPU Hardware Errors	2	U / M / M / M	Errors from RPU0 and RPU1 which includes both Correctable and Uncorrectable Errors.
RPU Lockstep Errors	2	M / M / M / M	RPU Lockstep Errors from R5_0 and R5_1. If the two bits are not 00, there is a Lockstep error. The Lockstep error is not initialized until RPU clock is enabled; therefore, error outcomes are masked by default and are expected to be unmasked after processor clock is enabled and before its reset is released.
RPU Temp Shutdown Alert	1	U / M / M / M	RPU Temperature Shutdown Alert from SYSMON.
APU Temp Shutdown Alert	1	U / M / M / M	APU Temperature Shutdown Alert from SYSMON.
RPU Common Cause Failure	1	U / M / M / M	RPU Common Cause Failures ORed together. The CCF Error register with the masking capability has to reside in the RPU.
LPD WDT Error	1	U / M / M / M	Error from Watchdog Timer in the LPD Subsystem.
FPD WDT Error	1	U / M / M / M	Error from Watchdog Timer in the FPD Subsystem.
Power Supply Failure Errors	8	U / M / M / M	Power Supply Failure Detection Errors from SYSMON. • VCCO_PSIO_2 MIO Bank 2 Supply Bit 7 • VCCO_PSIO_1 MIO Bank 1 Supply Bit 6 • VCCO_PSIO_0 MIO Bank 0 Supply Bit 5 • VCCO_PSIO_3 Dedicated IO Supply Bit 4 • VCCO_PSDDR DDRIO Supply Bit 3 • VCC_PSAUX Aux Supply 1.8V Bit 2 • VCC_PSINTFP PD Supply 0.85V Bit 1 • VCC_PSINTLP LPD Supply 0.85V Bit 0
LPD XMPU Isolation Error	1	U / M / M / M	OR of all LPD XMPU errors
FPD XMPU Isolation Error	1	U / M / M / M	OR of all FPD XMPU errors
Clock Monitor Error	1	U / M / M / M	Error from Clock Monitor logic

Table 4-2: **Errors Routed To Error Propagation Logic and Mask Default State *(Cont'd)***

Error Source	No. of Error Bits	Reset State of Masks (M=Masked, U=Unmasked) PS_ERROR_OUT, PMU Int, SRST, POR	Notes
LPD Bus Timeout Error	1	U / M / M / M	OR of all LPD Bus Timeout errors
FPD Bus Timeout Error	1	U / M / M / M	OR of all FPD Bus Timeout errors
Generic PL Errors	4	U / M / M / M	Generic PL errors communicated to PS
PLL Lock Errors	5	M / M / M / M	PLL Lock Errors. The error will be unmasked after the PLL is locked to alert when the PLL loses lock. • VPLL Bit 4 • DPLL Bit 3 • APLL Bit 2 • RPLL Bit 1 • IOPLL Bit 0
CSU Error	1	U / M / M / M	CSU Hardware Errors. Includes CSU ROM Validation Error.

Most of the PS_ERROR_OUT notifications are unmasked by default after an external POR (i.e. during device start-up). The only exceptions are made for the notifications that might occur when the system is not yet initialized (e.g. ECC uncorrectable errors can occur indefinitely as long as the memory is not initialized by the application), therefore it is necessary that the Application initializes the related logic/memory before unmasking the error notification. For further information see *PMU Error Handling and Propagation Logic* in Chapter 6 of UG1085 [Ref 4].

Requirements Allocated To the Zynq UltraScale+ MPSoC Application (Assumptions Of Use)

Overview

This chapter describes the safety requirements for the item's application executing on the Zynq UltraScale+ MPSoC (assumptions of use), stemming from the Zynq UltraScale+ MPSoC safety analysis, and the software safety mechanisms (i.e. software test libraries) implemented by Xilinx. The Zynq UltraScale+ MPSoC FMEDA Report presents the analysis results for a fixed configuration based on these two categories. The considerations associated with using and communicating with peripherals are further discussed in Appendix A, Safety Mechanisms for Peripherals Protection. Sections in this chapter are cross-referenced to Table B-1, *Software Safety Mechanisms and Assumptions of Use* which identifies recommendations, fault coverage for permanent/transient faults, active status at startup/run-time, and STL or AoU identification.

The general assumptions of this chapter on the application are that the item integrator will implement all of the assumptions of use pertinent to the application, use the STL pertinent to the application as described in the Zynq UltraScale+ MPSoC's *Software Safety User Guide* (UG1220) [Ref 8] and enable all hardware safety mechanisms. As such, the words *must*, *will* and *shall* within the descriptions of the assumptions of use, identify the measures that the item integrator is expected to apply in order to achieve the highest level of safe operation. The non application of any assumption of use, be it a deliberate choice or through erroneous design, might result in a lower level of safety. Deliberate non-application of any requirement is acceptable provided that the safety goal of the item is achieved.

Note on the FMEDA report: The present Safety Manual is being released together with a Microsoft Excel document titled *ZU+_FMEDA_report.xlsx*. The FMEDA Report contains the results of the FMEDA analysis performed on the biggest device of the Zynq UltraScale+ family (Die: ZU19EG, Package: FFVE1924), therefore reporting the worst case scenario in terms of Permanent and Transient FIT rate for the whole device, and Single Point Fault Metric for the Low Power Domain.

Note on the Temperature Profile: The Permanent FIT rates reported in the FMEDA report is calculated assuming an average junction temperature of T_J=85°C during the device lifetime. The Transient (i.e. Single Event Effect) FIT rate in the FMEDA report is calculated assuming an average junction temperature T_J=80°C during the device lifetime. According to the measurements taken by Xilinx, the Transient FIT rate does not have significant variation when varying the temperature profile, therefore no Transient FIT recalculation is needed in case the user estimates a different Temperature Profile. If the user estimates a greater average lifetime for T_J, then the Permanent FIT rate needs to be recalculated and the user should contact Xilinx customer support.

Assumptions of Use from the Safety Concept and the Safety Analyses

Note: Xilinx SM tracking tags indicated in this chapter are used by Xilinx to maintain requirement traceability. They should be ignored by the user.

Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols

Xilinx SM tracking tags: [SM_IOU_WDT] [SM_IOU_APP1] [SM_QSPI_APP1] [SM_QSPI_APP2] [SM_CAN_APP1] [SM_CAN_APP2] [SC_AOU017] [SC_AOU020] [SM_INT_LPD_APP]

Description

There are no error handling mechanisms implemented in the peripherals and interconnect located within the LPD. Any use of the following features will achieve diagnostic coverage through the application of data encapsulation and end-to-end protection protocols.

1. Multiplexed Input/Output (MIO)
2. Gigabit Ethernet MAC (GEM)
3. Not-AND Flash Memory Controller (NAND)
4. Secure Digital memory (SD3.0/SDIO/eMMC4.51) controller
5. Quad Serial Peripheral Interface (Quad-SPI) controller
6. Serial Peripheral Interface (SPI) controller
7. Controller Area Network (CAN) controller
8. Inter-Integrated Circuit (I2C) controller
9. Universal Asynchronous Receiver/Transmitter (UART)
10. Low Power Domain Direct Memory Access (LPD-DMA)
11. General Purpose Input/Output (GPIO)
12. LPD interconnect

Application software residing in the RPU operating in lockstep mode or the PMU triple redundant processor shall be used to encapsulate data employing a scheme that provides adequate coverage for the nature and quantity of data being transferred. The application software will encapsulate data before writing to any peripheral or memory. The application software will verify the encapsulated data when reading from any peripheral or memory and flag any error that should occur to the safety application. For example, frames consisting of a header, a fixed number of bytes and a 32-bit CRC might be used. The application software will calculate a CRC before writing a frame to any peripheral or memory. The application software will re-compute a CRC when reading a frame from any peripheral or memory and flag any CRC error that should occur to the safety application.

In cases where a data encapsulation is not practically implementable, a suitable combination of the following mechanisms shall be implemented:

- Data read-back when writing (i.e. data verification).
- Communication time-out management using an internal or external Watchdog timer.
- Testing that peripherals outside of the LPD wake-up every FTTI when in sleep mode is used.
- Watchdog to observe RPU behavior when executing code stored in OCM.
- Use of Software Test Library (STL) to periodically verify functionality.

Implementation

Figure 1-1 in UG1085 [Ref 4] defines the peripherals and interconnect within the LPD. The programming steps are architecture specific and depend on which features are used. The use of peripherals is further discussed in Appendix A, Safety Mechanisms for Peripherals Protection

RPU Verification of Interrupts

Xilinx SM tracking tags: [SM_RPU_APP1]

Description

As part of its interrupt handling routines, the RPU shall verify the integrity of every interrupt before initiating any actions. The verification processes implemented will require knowledge about the correct and anticipated nature of the system and exploit that knowledge in order to identify invalid interrupts. Schemes for consideration are as follows:

- The RPU may immediately reject an interrupt from a particular originator if it is not associated with the priority expected from the same originator. This can identify erroneous behavior without requiring further communication thereby reducing latency.
- On receiving an interrupt the RPU might communicate with the originator of the interrupt in the system to independently verify that the interrupt was intended. Ideally, the communication will be able to isolate and verify each interrupt through the use of unique information such as a time stamp or identification ensuring that each interrupt is processed only once.

- The RPU might immediately reject an interrupt from a particular originator if it occurs too soon after a previous interrupt from the same originator or other suitable reference. As well as identifying a continuous interrupt condition this can identify erroneous behavior without requiring further communication thereby reducing latency.
- The RPU might immediately reject an interrupt from a particular originator if it occurs too long after a previous interrupt from the same originator or other suitable reference. The RPU might also consider that an error has occurred if no interrupt is received from a given originator before a certain time limit has expired. These schemes can identify erroneous behavior without requiring further communication thereby reducing latency.
- Any interrupt that is not successfully verified or not received within an expected time will be flagged to the safety application without implementing any actions normally associated with it.

Implementation

- The inclusion of code specific to the known and expected operation of the system
- Potential use of timers

RPU Communication with GIC

Xilinx SM tracking tags: [SM_RPU_APP2]

Description

The real-time processing unit (RPU) communicates with the general interrupt controller (GIC) using a low latency peripheral port (LLPP).

The application software will verify every write made to the GIC. Each write to the GIC will be followed by a read from the GIC and a comparison of the two values. Any non-matching cases shall be flagged to the safety application.

The application software will verify every read from the GIC. Each read from the GIC will be repeated at least two times and the values compared. Any non-matching cases shall be flagged to the safety application.

Implementation

See Chapter 4 in UG1085 [Ref 4] for the RPU and Chapter 11 for the GIC. Figure 4 in UG1085 Chapter 4 illustrates the LLPP connection between the R5 processors, which are used in lock-step, and the GIC.

RPU AXI Communication

Xilinx SM tracking tags: [SM_RPU_APP3]

Description

The AXI ID compressor block (IDC) is required to compress the larger AXI ID used in the rest of the system to the 8-bit AXI ID supported by the Cortex R5.

The application software will verify every write made to the IDC. Each write to the IDC will be followed by a read from the IDC and the two values will be compared. Any non-matching cases shall be flagged to the safety application.

The application software will verify every read from the IDC. Each read from the IDC will be repeated at least two times and the values compared. Any non-matching cases shall be flagged to the safety application.

Implementation

With reference to Figure 4-1 in Chapter 4 of UG1085 [Ref 4], the IDC is located between the 64-bit paths between the switch following the XPPU and each R5 processor.

LPD-DMA Usage

Xilinx SM tracking tags: [SC_AOU007] [SC_AOU303]

Description

The 8-channel LPD-DMA is also known as the ADMA. The Flow-Control Interface (FCI) shall not be used when accessing the Programmable Logic (PL).

The first-in-first-out (FIFO) blocks also known as the common buffer, are implemented using standard memory cells and there is no hardware logic to detect any failure of the memory cells of the FIFO or the buffer descriptors.

Before initiating DMA *write* transactions, the application software residing in either the RPU operating in lockstep mode or the PMU triple redundant processor shall encapsulate the data into fix-sized frames each containing a header (or delimiter), a defined number of bytes and a computed CRC-32. Following DMA *read* transactions, the application software shall verify the format of each frame and re-compute the CRC for each frame and flag any errors that should occur to the safety application.

The application software will include an LPD-DMA device driver that is responsible for configuring the DMA. During the safety mission the configuration of the LPD-DMA shall be checked every FTTI and any errors flagged to the safety application. When the Scatter Gather DMA Mode is used the DMA controller's software (driver) shall implement a CRC computation of the DMA buffer descriptors that are stored in memory as a circular ring of descriptors. The controller's software shall assign a unique sequence number to each descriptor. It is the responsibility of the controller's software to verify order of completion of DMA descriptor based on the assigned sequence number.

Implementation

See Chapter 17 in UG1085 [Ref 4].

MBIST Usage

Xilinx SM tracking tags: [SC_AOU031]

Description

The PMU automatically initiates a memory built-in self-test (MBIST) during the boot phase following POR, and any errors reported must be flagged to the safety application. Prior to the start of the safety mission, it must be confirmed that MBIST has completed and then all memories that are to be used during the safety application must be initialized as required. MBIST shall not be used during the safety mission.

Implementation

Chapter 6 in UG1085 [Ref 4] describes MBIST which is performed by the PMU after POR and before release of reset to the CSU. The MBIST_DONE register contains 32 completion signals and the MBIST_GOOD register contains 32 status signals.

XMPU Configuration

Xilinx SM tracking tags: [SC_AOU008] [SC_AOU009] [SC_AOU011] [SC_AOU004] [SC_AOU010]

Description

The memory protection unit (XMPU) provides memory protection against unintended access from all masters including those implemented in the FPD and PL as well as those contained in the LPD.

During the start-up phase, the registers associated with the LPD XMPU shall be configured with pre-defined master-slave accesses by the FSBL which limit accesses to the minimum required in order to facilitate the boot and configuration process.

Following boot and configuration, but prior to the start of the safety mission, the registers associated with the LPD XMPU shall be configured as required for operation during the safety mission. The configuration of the LPD XMPU shall be one which prevents any AXI masters located in the PL and/or FPD (classified at a lower ASIL level) to access any part of the OCM located within the LPD. The configuration and integrity of the XMPU shall be checked using STL fault injection and any errors flagged to the safety application.

During the safety mission the configuration of the XMPU shall be checked every FTTI and any errors flagged to the safety application. Likewise, any permission violations detected and handled by the XMPU shall flagged to the safety application.

Implementation

The XMPU is described in the *System Protection Unit* section in Chapter 16 of UG1085 [Ref 4]. Figure 16-1 in UG1085 illustrates the multiple points at which the XMPU provides system-level protection.

XPPU Configuration

Xilinx SM tracking tags: [SC_AOU008] [SC_AOU009] [SC_AOU011] [SC_AOU004] [SC_AOU010]

Description

The peripheral protection unit (XPPU) provides LPD peripheral isolation and Inter-Processor Interconnect (IPI) protection against unintended access from all master including those implemented in the FPD and PL as well as those contained in the LPD.

During the start-up phase, the registers associated with the XPPU shall be configured with pre-defined master-slave accesses by the FSBL which limit accesses to the minimum required in order to facilitate the boot and configuration process.

Following boot and configuration, but prior to the start of the safety mission, the registers associated with the XPPU shall be configured as required for operation during the safety mission. The configuration and integrity of the XPPU shall be checked using STL fault injection and any errors flagged to the safety application.

Whenever possible, the configuration of the XPPU shall be one which prevents any AXI masters located in the PL and/or FPD (classified at a lower ASIL level) to access any peripheral located within the LPD. However, the XPPU can be configured to allow an AXI master within the FPD or PL to access a peripheral within the LPD provided the peripheral is then dedicated to only that master (i.e. is not shared with an AXI master located within the LPD). Any peripherals dedicated to an AXI master located in the PL and/or FPD shall be classified as being at the lower ASIL level.

During the safety mission the configuration of the XPPU shall be checked every FTTI and any errors flagged to the safety application. Likewise, any permission violations detected and handled by the XPPU shall flagged to the safety application.

Implementation

The XPPU is described in the *System Protection Unit* section in Chapter 14 of UG1085 [Ref 4]. Figures 4-1, 14-1, 33-4, 33-5 and 33-6 in UG1085 illustrate the multiple points at which the XPPU provides system-level protection.

RPU WDT and PMU WDT Configuration in LPD

Xilinx SM tracking tags: [SC_ AOU041]

Description

Watchdogs must be used to monitor the execution flow of the RPU and PMU. External watchdogs may be used (also see Clock Faults Detection) and/or the two system watchdog timers (SWDT) available in the LPD can be exploited. When the SWDT are used then the RPU SWDT shall be configured and utilized to check the RPU execution flow and the PMU SWDT shall be configured and utilized to check the PMU execution flow.

If the RPU fails to update the RPU SWDT, a timeout of RPU SWDT shall trigger a system restart or a RPU reset based on the configuration of the RPU SWDT module. The functionality of PMU SWDT is identical to RPU SWDT and can also be used to trigger a system restart or a PMU reset.

Logic built-in self-test (LBIST) shall be enabled during start-up to verify the functional integrity of the PMU SWDT. Any error on PMU SWDT hardware logic is flagged by LBIST and shall be treated as a critical error in safety applications.

The RPU SWDT is not verified by logic built-in-self-test (LBIST) during start-up. It is recommended that the application software verifies the integrity of the RPU SWDT as part of the start-up process.

Implementation

See Chapter 12 in UG1085 [Ref 4].

The SWDT Error Injection STL can be used to verify the integrity of either SWDT.

LBIST Permanently Enabled

Xilinx SM tracking tags: [SC_AOU030] [DFA_AOU001]

Description

The logic built-in self-test (LBIST) mechanism shall be permanently enabled through the appropriate setting of eFUSE. The eFUSE settings shall be confirmed prior to the start of the safety mission.

If LBIST detects a failure the PS_ERROR_STATUS pin is asserted and the PS will not boot. The system should monitor the PS_ERROR_STATUS pin and flag the failure to the Safety Application.

Implementation

See Chapter 37 in UG1085 [Ref 4] and UG1191 (contained within UG643 [Ref 14]). The PS DAP controller supports PS eFUSE programming. The LBIST_EN control is bit10 of the MISC_USER_CTRL register. In addition to reading the eFUSE settings, the activation of LBIST can also be confirmed through an observation of time to boot. The output of an Internal Ring Oscillator (IRO) with a frequency in the range 142 MHz to 220 MHz is divided by two and supplied to the LBIST circuitry. When LBIST is enabled it requires approximately 736,000 clock cycles to complete all tests thereby increasing boot-time by approximately 6.7 ms to 10.4 ms. Suitable observation points for a timing measurement are the POR and the signals associated with the quad-SPI flash memory. The time from release of POR to first quad-SPI activity is also dependent on the frequency of the IRO therefore time measurements should be made before and after programming of the relevant FUSE. Using the suggested observation points, a comparison of times measured should clearly show the additional delay after LBIST has been enabled (approximately 2 ms before and 10 ms after LBIST is enabled).

Power Supplies

Xilinx SM tracking tags: [SC_ AOU005] [SC_AOU003]

Description

All power rails supplied to the device must comply with the recommended operating conditions presented in Table 2 of the Zynq UltraScale+ MPSoC's Data Sheet (DS925) [Ref 5]. An external voltage monitor shall be implemented and monitored by the safety application.

The system monitor can be used to verify the measurements of the external monitor provided the low power domain (VCC_PSINTLP) required for the LPD and system monitor to operate are within recommended limits.

Implementation

External to the device. The external monitor can form part of the power supply controller (e.g. using PMBUS).

Integrity of the Configuration Registers

Xilinx SM tracking tags: [SC_ AOU019]

Description

Application software shall check the data integrity of all safety critical configuration registers in the LPD at least once every FTTI with any errors flagged to the safety application. The software test libraries (STL) relating to register coverage shall be used to create a golden copy of the safety critical registers (XStl_RegCheckInitTop) immediately following start-up and then to check all safety critical registers during the safety mission (XStl_RegCheckTop).

Implementation

See *Register Coverage* in Chapter 2 of UG1085 [Ref 8].

Triple-Timer Counters in LPD

Xilinx SM tracking tags: [SC_ AOU040]

Description

There are four triple timer counters (TTC) available as peripherals in the low power domain (LPD). Although each TTC contains three independent timer/counter modules and it should be recognized that a TTC does not provide a triple modular redundant (TMR) scheme. Therefore, safety aware application software requiring a timer shall employ a pair of TTCs and implement a dual redundancy scheme.

The application software shall employ two timer/counter modules, each of which are located in a different TTC. The application software will read and compare the output values from both counters at periodic intervals and flag any errors to the safety application.

When implementing dual redundant timer/counter modules the same reference clock source must be provided to both of the TTCs in which they are located.

Implementation

See Chapter 12 in UG1085 [Ref 4].

System Monitors

Xilinx SM tracking tags: [SC_ AOU018] [SM_AMS_APP1] [SM_AMS_APP2] [SM_AMS_APP3] [SM_AMS_LOG1]

Description

The RPU operating in lockstep mode shall use the system monitor within the LPD (PS-SYSMON) to monitor the internal supply voltages associated with the LPD and die temperatures associated with the LPD and FPD. The PS-SYSMON shall be configured to enable the channel sequencer to perform continuous autonomous sampling, limits checking and interrupt generation. The configuration of sample rate, sample averaging, and channels to be sequenced shall be defined such that the update time of the status registers within the PS-SYSMON do not exceed one third of the FTTI. The upper and lower voltage thresholds corresponding with each LPD supply voltage channel shall be defined to be at or within the recommended operation conditions of the device. The over-temperature (OT) threshold corresponding with the LPD and FPD temperature channels shall be set at or below the operational limits of the device and the regular temperature threshold should be set at or below the OT thresholds.

The interrupt masks defined by the IMR_0 and IMR_1 registers shall be configured to generate AMS Interrupt 88 to the RPU when the PS-SYSMON generates an alarm associated with any LPD supply voltage or any LPD or FPD temperature channel exceeding a threshold. On receiving an AMS interrupt, the RPU shall first read the ISR_0 and ISR_1 registers to determine the source and then read the status register corresponding to the source to establish its current value. The safety application shall be informed of any parameter that exceeds limits and its current value. If the LPD or FPD temperature exceeds the over-temperature (OT) limit the safety application shall turn off all power supplies to the device and allow adequate time for the device to cool and for all of the power rails to fall below 0.1v before attempting to re-start. If the LPD or FPD temperature exceeds the regular temperature threshold, a thermal management and/or monitoring scheme can be initiated. If any LPD voltage limit is exceeded, the corresponding power supply shall be adjusted to maintain operation within the recommended conditions of the device or all power supplies to the device shall be turned off.

PS-SYSMON is provided to enhance overall system integrity but does not contain any hardware logic to detect failures within the block itself. Application software residing in the RPU operating in lockstep mode or the PMU triple redundant processor shall verify the operation of the PS-SYSMON during the start-up phase and every FTTI during the safety mission.

As is not possible to implement a data encapsulation scheme when using the PS-SYSMON peripheral, values written to configuration registers shall be read back and verified. Following an AMS interrupt, the RPU will read the corresponding status register and independently verify the value which the autonomous channel sequencer used to generate the alarm and interrupt in the first place.

During the start-up phase the integrity of the analogue-to-digital converter within the PS-SYSMON shall be confirmed. At least one parameter that can be independently verified to be within tolerance shall be read from the appropriate status register within the PS-SYSMON. Suitable candidates are a fixed reference voltage or an LPD power supply whose level is known using an independent external power supply watchdog. The configuration and operation of the continuous autonomous sampling circuits shall be verified by reading the corresponding status registers for each of the sequenced channels and confirming that each parameter has a value within its expected limits.

During the safety mission the RPU or PMU shall read the status registers corresponding with the LPD and FPD temperature channels three times during each FTTI. The intervals between each of the three read operations shall not be less than the update time of the status registers within the PS-SYSMON. The RPU or PMU shall independently verify that LPD and FPD temperature values fall above the lower operational limit of the device and below the over-temperature (OT) threshold. Using the full 16-bit values read from the LPD temperature status register, each set of three values shall be compared to establish if they are identical or if there is any variation. Likewise, using the full 16-bit values read from the

FPD temperature status register, each set of three values shall be compared establish if they are identical or if there is any variation. If the LPD and/or FPD temperature falls outside of operational limits or any set of three temperature values are identical then the safety application shall be informed and the safety application shall turn off all power supplies to the device and allow adequate time for the device to cool and for all of the power rails to fall below 0.1v before attempting to re-start.

The system monitor located within the programmable logic domain (PL-SYSMON) is capable of monitoring the supply voltages and temperature associated with the PL and up to 17 external analogue inputs. The PL-SYSMON can either be treated as a peripheral to the processing system (PS) or as a core within the hardware design implemented in the programmable logic (PL). Various conditions need to be satisfied for either the PS or PL to be able to access the PL-SYSMON. When conditions permit the PL-SYSMON to be accessed by the PS or the PL during the safety mission then the PL temperature channel shall be monitored. The status register corresponding with the temperature channel shall be read at intervals exceeding the update time of the PL-SYSMON status registers by the channel sequencer and not less than once per second. Each temperature value shall be verified to be between the lower and upper operational limits of the device. Using the full 16-bit values read from the temperature status register, each set of three values shall be compared to establish if they are identical or if there is any variation. If the PL temperature falls outside of operational limits of the device or any set of three temperature values are identical then the safety application shall be informed and the safety application shall turn off all power supplies to the device and allow adequate time for the device to cool and for all the power rails to fall below 0.1v before attempting to re-start.

Implementation

Before the RPU attempts to access the PS-SYSMON it must check the jtag_locked bit in the MON_STATUS (AMS) register to confirm that a valid clock is being provided to the APB interface which is used to access the PS-SYSMON. The configuration registers must be set to define the sample rate, the channels to be sequenced (which must include all LPD supply voltage channels, and the LPD and FPD temperatures) and the amount of sample averaging. The combination of these settings will determine the rate at which the status registers will be updated and needs to be less that FTTI/3. The corresponding alarms and associated interrupts for each sequenced channel must also be enabled.

The System Monitor in the PL (PL-SYSMON) can either be accessed by a processor in the PS or by the design in the PL. If the PL is configured with a design containing the SYSMONE4 primitive then only the PL design can access the PL-SYSMON and the safety requirements associated with the PL-SYSMON will need to be implemented as part of the PL design. If the PL is not configured or configured with a design that does not contain the SYSMONE4 primitive then a processor in the PS has the potential to use the PL-SYSMON as a peripheral. If the RPU is used to control and monitor the PL-SYSMON via the APB interface it must first check the jtag_locked bit in the MON_STATUS (AMS) register to confirm that a valid clock is being provided to the APB interface and check the accessible bit in the PL_SYSMON_CONTROL_STATUS (AMS) register to confirm that the required power supplies are active, the PCAP isolation is inactive, and that the PL is not configured with a design containing the SYSMONE4 primitive. Furthermore, to avoid non-deterministic behavior, any conflict with the JTAG and I2C/PMBus communication with PL-SYSMON must be avoided. The JTAG and I2C/PMBus interfaces are both available for use prior to PL configuration so the RPU should only attempt to access the PL-SYSMON if neither of these external interfaces can be used. Dedicated access to the PL-SYSMON via the APB interface can be guaranteed following configuration of the PL with a bit stream using the `set_property BITSTREAM.GENERAL.JTAG_SYSMON DISABLE [current_design]` option. Configuration of the PL always disables the I2C/PMBus interface. Regardless of which interface is used to access the control and status registers within the PL-SYSMON, its alarm signals are permanently connected to the PS via the PCAP so as long as PCAP isolation is inactive the PL-SYSMON alarms can be observed by the PS reading the ISR_0 (AMS) and these alarms can contribute to the generation of interrupts.

The analog-to-digital converter within each System Monitor generates 16-bit samples. The specified resolution of 10-bits relates to the 10 most significant bits of each 16-bit sample. The 6 least significant bits of each sample can be used to minimize the quantization effects or improve resolution through averaging. Regardless of the degree of averaging employed, the status registers hold 16-bit values. When analyzing sets of three value temperature values all 16-bits must be compared.

Measurements of LPD, FPD, and PL supply voltages can also be used to contribute to the control of power supplies. Fine tuning of the power supply voltages can help prevent limits being exceeded and to operate the device at lower valid voltages, reducing power dissipation. Recording the minimum and maximum values captured by the System Monitors might reveal supply voltage drift over time due to environmental conditions and/or aging and these could be used to preempt issues that can be proactively addressed at a suitable product service intervals.

Each temperature sensor is associated with a pair of upper and lower limits (thresholds). The regular limits and associated alarm are suitable for the implementation of device cooling schemes (e.g. cooling fan and/or load reduction) that maintain operation at a lower temperature. The over-temperature (OT) limits and associated alarm are typically set at or close to the maximum recommended operating temperature of the device. The OT alarm would be used to trigger an immediate but controlled shut down of the equipment before erroneous operation or permanent damage might occur.

See System Monitor chapter in UG1085 [Ref 4] and *UltraScale Architecture System Monitor User Guide* (UG580) [Ref 6].

Quad-SPI for Boot and Configuration

Xilinx SM tracking tags: [SC_AOU021]

Description

The quad serial peripheral interface (quad-SPI) boot mode shall be used to boot the device from 24 or 32-bit quad-SPI flash. The quad-SPI 32-bit boot mode shall be used for flash sizes larger than 16 MB and with any quad-SPI flash that supports 32-bit addressing.

Implementation

See Chapter 9 in UG1085 [Ref 4].

Quad-SPI Controller Protection

Xilinx SM tracking tags: [SC_AOU022]

Description

Quad-SPI flash used as the boot device can also be used for non-volatile storage of media. Using quad-SPI with the secure boot mode the SHA-3 algorithm produces a 384-bit digest that authenticates the first stage boot loader (FSBL) image.

In all other situations where quad-SPI is used there are no error handling mechanisms implemented in the hardware. Application software residing in the RPU operating in lockstep mode or the PMU triple redundant processor shall be used to encapsulate data employing a scheme that provides adequate coverage for the nature and quantity of data being stored. The application software will encapsulate data before writing to memory. The application software will verify the encapsulated data when reading from memory and flag any error that should occur to the safety application. For example, frames consisting of a header, a fixed number of bytes and a 32-bit CRC might be used. The application software will calculate a CRC before writing a frame into memory. The application software will re-compute a CRC when reading a frame from memory and flag any CRC error that should occur to the safety application.

If using the quad-SPI DMA for transferring the data from quad-SPI flash memory to DDR memory, the application software will include a device driver that is responsible for configuring the quad-SPI DMA and will periodically confirm the configuration. It is necessary to protect the quad-SPI DMA buffer descriptor with a descriptor CRC which needs to be computed by the device driver. CRC-16 can be performed for buffer descriptors. Each buffer descriptor shall be assigned with a sequence number. While processing the buffer descriptor, the device driver shall verify order of completion of the buffer descriptors based on the sequence number.

Implementation

See Chapters 9, 10, and 22 in UG1085 [Ref 4].

Non-safety Related Modules and Features in LPD

Xilinx SM tracking tags: [MRD_AOU023] [SC_AOU042]

Description

Any features in the LPD that are never used by the application must be disabled and the configuration that disables these unused features shall be checked every FTTI with any errors flagged to the safety application. All disabled modules can be classified as being non-safety related modules.

The following modules contained in the LPD can be used to initialize, service, and support the application but shall never be used during the time when the application is actively performing the safety mission:

- AXI trace monitor (ATM)
- AXI performance monitor (APM)
- Battery backed random access memory (BBRAM)
- Real time clock (RTC)
- JTAG
- ARM debug access port (ARM DAP)
- Crypto interface block (CIB)

The configuration that disables all of these features shall be checked every FTTI and any errors flagged to the safety application.

Implementation

See Chapters 6, 10, and 37 in UG1085 [Ref 4] and *Register Coverage* in Chapter 2 of UG1220 [Ref 8].

Non-safety Interfaces

Xilinx SM tracking tags: [MRD_AOU023] [SC_AOU042]

Description

Due to their reliance on the full power domain (FPD) and the Cache Coherent Interconnect (CCI), the following interfaces must not be used by the LPD during the time when the application is actively performing the safety mission:

- High performance PL master AXI interface (HPC)
- High performance AXI interface (AXI-HP)
- High performance PS master AXI interface (HPM)

The configuration that disables the LPD from using these interfaced shall be set before the start of the safety mission and checked every FTTI with any errors flagged to the safety application.

These interfaces can be used by the LPD to initialize, service, and support the application prior to the safety mission.

Note: This assumption of use is specifically related to the use of these interfaces by the LPD. The FPD (classified at a lower ASIL level) can use these interfaces at any time.

Implementation

See Chapter 13 in UG1085 [Ref 4] and *Register Coverage* in Chapter 2 of UG1220 [Ref 8].

Dynamic Power Management in LPD

Xilinx SM tracking tags: [SC_AOU015]

Description

Dynamic power management shall not be used in the LPD. The RPU shall be operated in *Run* mode and the PMU will remain in *Power Up* mode at all times. During start up the peripherals will be configured as required and continue to operate in the power modes assigned to them. This configuration shall be checked every FTTI and any errors flagged to the safety application.

Implementation

See the *Power Management Modes* table in Chapter 4 of UG1085 [Ref 4] for the RPU, and Chapter 6 in UG1085 for the PMU.

Dynamic Power Management in FPD and PL

Xilinx SM tracking tags: [SM_INT_APP1]

Description

Power management of the full power domain (FPD) or the Programmable Logic domain (PL), either partially or entirely, must only be performed in ways that are compatible with maintaining the overall safety goal of the item.

Power consumption can be controlled using any recognized technique including adjustments to clock frequencies, asserting resets, enabling/disabling features, and powering down of individual features or whole domains. Regardless of technique, the master that invokes the power management shall be located within the LPD (i.e., the PMU or the RPU). The same master must monitor the transition between power states and verify that the process has successfully completed. Furthermore, the same master shall verify that the power states of all other domains and features has remained unaltered. Any errors shall be flagged to the safety application.

Implementation

Power down and power up requests are made to the PMU by writing to the REQ_PWRDWN_TRIG and REQ_PWRUP_TRIG registers. The status of the interrupt signals presented to the PMU can be verified by reading the REQ_PWRDWN_STATUS and REQ_PWRUP_STATUS registers. Fields of the status registers containing `1` indicate that a request is pending. Once a request has been serviced the corresponding field will contain `0`. A request can also be canceled by the timely writing of a `1` to the corresponding field of the status registers. The fields of the PWR_STATE register confirm the powered down (0) or powered up (1) states of each island.

Power management techniques that simply control features during operation will involve writing to associated registers and then verifying those settings and their effects. The application should take care to ensure that deliberate modifications to the contents of control registers for power management purposes does not confuse integrity checks of the same. Likewise, clock monitors should be adjusted to verify deliberate adjustments to clock frequencies or the effects of powering down of a region.

DAP to be Turned Off

Xilinx SM tracking tags: [SC_AOU016]

Description

The Debug Access Port (DAP) in the LPD must not be used at any time and must be turned off. The configuration that disables DAP shall be checked every FTTI with any errors flagged to the safety application.

Implementation

The “ssss_rpu_dbgen” bit in the JTAG_DAP_CFG register of the CSU defines the state of the DBGEN signal input to the RPU. When this signal is Low debug is disabled regardless of the values of the MDBGen and HDBGen bits in the DBGDSCR resister of the RPU.

The “ssss_rpu_ niden” bit in the JTAG_DAP_CFG register of the CSU defines the state of the NIDEN signal input to the RPU. This signal can be driven High to facilitate monitoring of RPU memory ECC errors using the RPU_0_ISR and RPU_1_ISR registers. The NIDEN signal only enables the non-invasive debug capabilities of the processor and does not put the processor into debug mode.

USB

Xilinx SM tracking tags: [SC_AOU016]

Description

The default configuration of the Universal Serial Bus (USB) Controller shall be one in which it is unused and turned off (see the following note). The configuration that disables USB shall be checked every FTTI with any errors flagged to the safety application.

Note: The USB can be used provided the XPPU is configured to dedicate the USB controller to an AXI master within the FPD or PL (classified at a lower ASIL level) and that it is not used to communicate any safety-related information. It will be recognized that USB 3.0 also requires the PS-GTR block which is located within the FPD.

Implementation

The processor configuration wizard (PCW) can be used to generate a configuration file (psu_init) in which the USB is disabled.

PMU Processor Usage

Xilinx SM tracking tags: [SC_AOU013] [SC_AOU014]

Description

The platform management unit (PMU) is a triple redundant processor. This processor shall only control the power-up, reset and monitoring of resources within the entire device. Usage of the PMU can include the capture, processing and management of error signals routed to it and the assertion of the dedicated PS_ERROR_OUT pin for observation by the external safety application.

At no time must the PMU be used to directly implement any functions forming the safety application and the PMU shall not handle interrupts originating outside the LPD. Furthermore, no AXI Masters, including AXI Masters located within the LPD, shall have write access to the PMU.

Implementation

See Chapter 6 in UG1085 [Ref 4]. The PMU will need to monitor or poll the FPD/PL to observe a request to power down.

Clock Faults Detection

Xilinx SM tracking tags: [SC_AOU006] [SM_CRL_LOG4] [SM_CRL_APPEXWDT] [SC_AOU400]

Description

The integrity of all clocks used within the Low Power Domain (LPD) must be confirmed.

There are eight Clock Monitors (CLKMON) provided in the LPD. Each Clock Monitor can be assigned to monitor any of the eight clocks within the LPD and evaluate the frequency of a selected clock relative to a choice of two reference clocks. Each Clock Monitor can be configured to generate an interrupt if the frequency of the monitored clock deviates outside a user defined range. The Clock Monitors must be used to monitor each of the used clocks within the LPD and any faults shall be flagged to the Safety Application. Prior to the start of the safety mission and once every FTTI during the safety mission the ability of the Clock Monitors to detect clock faults and to generate interrupts shall be verified.

The PMU is clocked by iro_clk which is generated by an oscillator within the LPD. A Clock Monitor using ps_ref_clk as the reference must be used to verify that the ro_clk is within the frequency range of 146 MHz to 220 MHz. An interrupt generated by the Clock Monitor assigned to this clock must be handled by the RPU which must flag any faults to the Safety Application.

The Clock Monitors do not monitor duty cycle, jitter, or quality of a clock. Furthermore, the Clock Monitors will only be able to report faults if lpd_lsbus_clk (derived from ps_ref_clk) is active. As a result, the Clock Monitors cannot be used to detect complete failures of ps_ref_clk or lpd_lsbus_clk. In order to detect major failures of these clocks, an independent watchdog that is located outside of the Low Power Domain (LPD) must be provided to monitor operation of the RPU and PMU within the LPD. Code executed by the RPU and PMU shall generate heartbeats with recognizable non-static signatures that are repeated at least twice every FTTI. The external watchdog shall observe the heartbeats and flag an error to the safety application if a valid heartbeat fails to be observed within an acceptable time window.

Implementation

The independent watchdog can be implemented in the PL or external to the Zynq UltraScale+ MPSoC device. The purpose of the scheme implemented is to detect major failures of the clocks within the LPD including a complete failure of the ps_ref_clock supplied to the device.

Clock Monitors are described in Chapter 37 of UG1085 [Ref 4]. The Clock Monitors enable the safety application to monitor and check the frequency of each clock to be verified. The functionality of each Clock Monitor and associated interrupt can be verified by temporarily modifying the upper and/or lower frequency thresholds such that the otherwise valid clock being monitored no longer falls within defined frequency limits. The SM_STL_CLK software test library can be used to inject errors using this method.

RPU CPU Configuration

Xilinx SM tracking tags: [SC_AOU043]

Description

The real time processing unit (RPU) contains a pair of Cortex-R5 processors and must be configured to operate in the locked mode. This is also known as *lock-step* operation or *safety mode*. The locked mode shall be enabled by the execution of suitable code during the start-up phase of the RPU.

Implementation

See *Lock-step Sequence in Cortex-R5 Processors* in Chapter 6 0f UG1085 [Ref 4]

Software Safety Mechanisms (Software Test Libraries)

Table 5-1 identifies software test libraries released by Xilinx. The first column lists the STL tags stemming from the Zynq UltraScale+ MPSoC FMEDA. This means that for the safety analysis, it has been assumed that a number of software test libraries are present and launched by the user application every FTTI. The second column identifies the STL actually developed and delivered by Xilinx. Therefore the table gives the correspondence between the safety analysis assumptions of the STL implementation and the actual STL software.

The User should ignore the tags in the first column, used for Xilinx internal traceability only. The description of the STL software and how the user application should integrate them is given in the Zynq UltraScale+ MPSoC's *Software Safety Safety User Guide* (UG1220) [Ref 8].

Note: According to analysis by Xilinx, each STL can be classified either as *highly recommended*, *recommended*, or *optional*. This classification can be found in Appendix B, List of Software Safety Mechanisms and Assumptions Of Use.

Table 5-1: **Software Test Libraries Released by Xilinx**

STL Tags from Safety Concept or FMEDA (Xilinx internal use, user should Ignore this)	SW Test Libraries Released by Xilinx
SM_RPU_STL1	SM_STL_GIC
SM_RPU_STL3 SM_PMU_STL4 SM_INT_LPD_STL1 SM_INT_LPD_STL2	SM_STL_INTC
SM_CRL_STL4	SM_STL_ATB
SC_REQ_326	SM_STL_MEM

Table 5-1: **Software Test Libraries Released by Xilinx *(Cont'd)***

STL Tags from Safety Concept or FMEDA (Xilinx internal use, user should Ignore this)	SW Test Libraries Released by Xilinx
SM_CRL_STL3	SM_STL_CLK
SM_RPU_STL2 SM_RPU_STL4 SM_PMU_STL1 SM_PMU_STL2 SM_PMU_STL3 (tbc) SM_PMU_STL4 SM_CSU_STL1 (tbc) SM_CRL_STL1 SM_CRL_STL2 SM_INT_LPD_STL1 SM_INT_LPD_STL2 SM_OCMADMA_STL1 SM_OCMADMA_STL2 SM_OCMADMA_STL3 SM_IOU_STL1 (tbc)	SM_STL_REG

Assumptions on STL Integration by the Application

The PMU is designed for Zynq UltraScale+ MPSoC services so it is recommended that the PMU should be used to run the STLs whenever possible. This in turn will free more of the RPU's processing power to run the user's application which is consistent with the general assumption of use for the Zynq UltraScale+ MPSoC. Xilinx SM tracking tag: [SC_AOU300]

If the elapsed time between two system start-ups is greater than 10 hours it is assumed that the user application will continuously execute the memory scrubbing routine SM_STL_MEM during mission. Xilinx SM tracking tags: [SC_AOU301]

Hardware Safety Mechanisms

This section identifies hardware mechanisms in the Low Power Domain (LPD) which are included specifically to enhance safety through fault mitigation and fault coverage. Hardware safety mechanisms are either permanently active or must be enabled by the application using the control registers indicated in order to achieve the highest level of safe operation.

Some hardware safety mechanisms automatically mitigate faults while others are primarily activity monitors whose detection and reporting of errors enhance fault coverage. To achieve the highest level of safe operation all fault reporting schemes indicated should be used to flag any errors to the safety application. It might be acceptable to temporarily or permanently ignore certain fault reports provided the overall safety goal of the item is achieved. For example, it might be deemed acceptable to ignore the report associated with the detection of a single bit upset to memory contents when ECC protection has been enabled to correct the read value before it is used.

Note: Xilinx SM tracking tags indicated in this section are used by Xilinx to maintain requirement traceability. They should be ignored by the user.

ECC Protection of Memories

Xilinx SM tracking tags: [SM_RPU_DFT] [SM_RPU_DFT2] [SM_RPU_LOG2] [SM_CRL_DFT1] [SM_PMU_DFT4]

Error Correcting Codes (ECC) and associated hardware circuits provide Single Error Correction and Double Error Detection (SECDED) for bit errors within words stored in memories within the LPD. Furthermore, the memory cells of different words are physically interleaved such that Single Event Upsets (SEU) that cause multiple bit upsets (MBU) have a high probability of resulting in two or more separate words each of which contain only a single bit error which can be detected and corrected. Although ECC is primarily associated with soft temporary errors caused by radiation (SEU) the same detection and mitigation also enhances coverage of hardware failures.

The PMU RAM has permanently enabled ECC protection. An un-correctable error is first reported by the N_FFail-RAM_UE and R_FFail-RAM_UE bits in the MB_FAULT_STATUS register. This is treated as a fatal error resulting in the automatic assertion of reset to the PMU. The default settings of the ERROR_EN_2 and ERROR_SIG_MASK_2 registers further propagate the un-correctable report through the PMU_UC bit of the ERROR_STATUS_2 register and assert the PS_ERROR_OUT pin. Correctable errors can be monitored by servicing interrupt 16 to the PMU.

The CSU RAM has permanently enabled ECC protection. An un-correctable error is first reported by the N_FFail-RAM_UE and R_FFail-RAM_UE bits in the CSU_FT_STATUS register. This is treated as a fatal error resulting in the automatic assertion and release of a reset to the CSU. This CSU restart will mitigate temporary soft errors caused by SEU albeit resulting in an increased overall execution time that maybe observed by an external watchdog. Any fatal error of the CSU, including an un-correctable ECC error, will also be recorded by the TMR_FATAL bit in the CSU_ISR register along with correctable errors reported by the CSU_RAM_ECC_ERROR bit. The CSU_IMR register can be configured to generate Interrupt 117 to the RPU and APU to monitor these status bits during operation.

The On-Chip Memory (OCM) requires ECC protection to be enabled by setting the ECC_ON_OFF bit in the OCM_ECC_CTRL register. The reset state of the OCM_ECC_CTRL register following a reset disables ECC but the ECC_ON_OFF bit is set by the CSU prior to the FSBL such that ECC protection is enabled and remains enabled unless subsequent user code deliberately disables it. The DET_ERROR bit in the OCM_ECC_CTRL register has a default value of `0` such that single bit errors are automatically corrected. Uncorrectable errors are reported by the ECC_EU and ECC_RMW bits in the OCM_ISR register along with correctable errors reported by the ECC_CE bit. The OCM_IMR register can be configured to generate Interrupt 42 to the RPU and APU. The ERROR_EN_1 register can be configured to propagate the ECC_EU error through to the OCM_ECC bit of the ERROR_STATUS_1 register. The

ERROR_INT_MASK_1, ERROR_SIG_MASK_1, ERROR_POR_MASK_1, and ERROR_SRST_MASK_1 registers can be configured to further propagate or process the error as desired. When accessing the OCM via the AXI slave interface, an un-correctable error results in a SLVERR if the UE_RES bit of the OCM_ERR_CTRL register is set (default value 0).

The two R5 processors forming the RPU have local cache memories and Tightly Coupled Memory (TCM). Within each R5, bits within the Auxiliary Control register (ACTLR) enable ECC protection. The TCM ECC is enabled by the B0TCMCEN, B1TCMCEN, and ATCMCEN bits and the cache ECC is enabled by setting three CEC bits. ECC is disabled following a reset but the default PMU firmware (PMUFW) enables ECC protection. Provided the "ssss_rpu_niden" bit in the JTAG_DAP_CFG register of the CSU and the "X" bit (bit 4) of the PMCR register inside the RPU are both set the reports of ECC errors associated with the memories or each R5 processor are initially reported in the RPU_0_ISR and RPU_1_ISR registers. Each register contains a total of 18 status bits associated with ECC covering un-correctable (or fatal) errors (UE or FAT) and correctable errors (CE). Status bits DTAG_DIRTY_CE, IDATA_CE, DDATA_CE, ITAG_CE, DTAG_DIRTY_FAT, and DDATA_FAT are related to the cache and status bits B0TCM_CE, B0TCM_UE, B1TCM_CE, B1TCM_UE, ATCM_CE, and ATCM_UE are directly related to the TCM. Status bits TCM_ASLV_CE, TCM_ASLV_FAT, TCM_LST_CE, TCM_LST_FAT, TCM_PREFETCH_CE, and TCM_PREFETCH_FAT identify the AXI slave, Load/Store or Pre-fetch interface in use when the TCM error was encountered. The RPU_0_IMR, RPU_1_IMR and ERROR_EN_1 registers can be configured to propagate RPU0 and RPU1 errors through to the ERROR_STATUS_1 register. The ERROR_INT_MASK_1, ERROR_SIG_MASK_1, ERROR_POR_MASK_1, and ERROR_SRST_MASK_1 can be configured to further propagate or process the error as desired. The RPU_0_IMR, RPU_1_IMR registers can also be configured generate Interrupts 44 and 45 to the RPU and APU.

Triple Modular Redundancy (TMR) Circuitry

Xilinx SM tracking tags: [SM_PMU_LOG1] [SM_PMU_LOG6] [SM_CSU_LOG1] [SM_CSU_LOG6] [SM_CRL_LOG1]

The implementation of circuits using the TMR method greatly increases the ability of a system to continue operating normally in the event of a soft or hardware failure. The fundamental premise being that if one out of three modules fails in some way, a voter observing the outputs from the three modules will allow the system to continue operating in a Dual Modular Redundant (DMR) mode using only the modules generating matching pairs of results. The mismatching results identify the failing module with the potential to initiate a procedure to attempt a full recovery to TMR mode following a soft error. Normal operation will only cease when mismatching results from all modules indicating two or three modules have failed.

The PMU processor and PMU associated control registers within the LPD are implemented using TMR with physical diversity and separation. The TMR technique will generally enable the PMU to continue operating normally but detection of module failures by the voter are reported in the MB_FAULT_STATUS register. The three N_LSFail and three R_LSFail status bits indicate lockstep mismatches between pairs of processors (modules). For example, if module 3 fails then modules 1 and 2 will continue to match but lockstep error status will be asserted between modules 1 and 3 and modules 2 and 3. There are also three N_FFail and three R_FFail status bits to indicate lockstep mismatches between the state machines associated with each processor (module) which report is the same way. Inspection of the MB_FAULT_STATUS will reveal the TMR status of the PMU.

The CSU processor and CSU associated control registers within the LPD are implemented using TMR with physical diversity and separation. The TMR technique will generally enable the CSU to continue operating normally but detection of module failures by the voter are reported in the CSU_FT_STATUS register. The N_MISMATCH_12_B, N_MISMATCH_13_B, and N_MISMATCH_23_B status bits and the R_MISMATCH_12_B, R_MISMATCH_13_B, and R_MISMATCH_23_B status bits indicate lockstep mismatches between pairs of processors (modules). The N_MISMATCH_12_A, N_MISMATCH_13_A, and N_MISMATCH_23_A status bits and the R_MISMATCH_12_A, R_MISMATCH_13_A, and R_MISMATCH_23_A status bits lockstep mismatches between the state machines associated with each processor (module). Inspection of the CSU_FT_STATUS will reveal the TMR status of the CSU.

Circuits associated with the generation of clocks and resets to the various elements are critical to operation and have been implemented using TMR with physical diversity and separation. The clock and reset logic (CRL) is tested as part of LBIST with the detection of any failures leading to assertion of the 'PS_ERROR_STATUS' pin and the PS will not boot.

Dual Modular Redundancy (DMR) Circuitry

Xilinx SM tracking tags: [SM_PMU_LOG2] [SM_PMU_LOG3] [SM_PMU_LOG4] [SM_CSU_LOG2] [SM_CSU_LOG3] [SM_CSU_LOG4] [SM_RPU_LOG1] [SM_RPU_LOG4] [SM_eFU_LOG1] [SM_CRL_DFT2]

The implementation of circuits using a DMR method is mainly employed to enhance detection of failures and hence improve fault coverage. The fundamental premise being that if the outputs of two modules do not match then one or both of the modules has failed in some way. However, there are a few cases in which DMR can be used to enhance reliability and availability.

The PMU is implemented using TMR which significantly increases the ability of the PMU to continue operating in the presence of a module failure. However, the TMR scheme places a dependence on the voter that compares the outputs from the three modules so DMR voters have been implemented. There is a DMR voter for the processor (module) outputs and a DMR voter for the outputs from the state machines associated with each processor. In each case, the pair of voters are referred to as being the Nominal (N) voter and the Redundant (R) voter. The detection of a voter mismatch is first reported in the MB_FAULT_STATUS register. Assertion of any one of the status bits N_FFail bits 8, 11, 12, 13, and 14, or R_FFail bits 8, 11, 12, 13, and 14 would indicate a fault condition. This is treated as a fatal error resulting in the automatic assertion of reset to the PMU. The default settings of the ERROR_EN_2 and ERROR_SIG_MASK_2 registers will further propagate the report through the PMU_UC bit of the ERROR_STATUS_2 register and assert the PS_ERROR_OUT pin.

The CSU is implemented using TMR which significantly increases the ability of the CSU to continue operating in the presence of a module failure. However, the TMR scheme places a dependence on the voter that compares the outputs from the three modules so DMR voters have been implemented. There is a DMR voter for the processor (module) outputs and a DMR voter for the outputs from the state machines associated with each processor. In each case, the pair of voters are referred to as being the Nominal (N) voter and the Redundant (R) voter. The detection of a voter mismatch is first reported in the CSU_FT_STATUS register. Assertion of any one of the status bits N_FT_ST_MISMATCH, N_COMP_ERR_12, N_COMP_ERR_13, N_COMP_ERR_23, N_VOTER_ERROR, and status bits R_FT_ST_MISMATCH, R_COMP_ERR_12, R_COMP_ERR_13, R_COMP_ERR_23, R_VOTER_ERROR would indicate a fault condition. This is treated as a fatal error resulting in the automatic assertion and release of a reset to the CSU. This CSU restart will mitigate temporary soft errors however caused albeit resulting in an increased overall execution time that maybe observed by an external watchdog. Any fatal error of the CSU including an un-correctable ECC error will also be recorded by the TMR_FATAL bit in the CSU_ISR register along with correctable errors reported by the CSU_RAM_ECC_ERROR bit. The CSU_IMR register can be configured to generate Interrupt 117 to the RPU and APU to monitor these status bits during operation.

When the two R5 processors forming the RPU are configured to operate in lockstep mode it becomes a DMR processor implementation with a voter. This voter is also a DMR implementation resulting in two lockstep (LS) status bits. If both voter status bits are asserted then both voters have detected a failure of one or both processors. A failure of a voter is indicated by a difference in the status bits (i.e., only one status bit will be asserted). The ERROR_EN_1 register can be configured to propagate the two bit RPU_LS status bits through to the ERROR_STATUS_1 register. ERROR_INT_MASK_1, ERROR_SIG_MASK_1, ERROR_POR_MASK_1, and ERROR_SRST_MASK_1 can be configured to further propagate or process the error as desired.

eFUSEs are one-time programmable cells that when programmed, should by definition permanently result in a static logic state. To increase reliability, each one-time programmable cell has been physically implemented using a DMR pair of eFUSEs. An actual failure will only occur if both the nominal and redundant fuses have failed. Failure of both eFUSEs will be reflected by unintended behavior of the function associated with the failed eFUSE (e.g., an issue with security resulting from incorrect key value).

The Logic Built-In Self-Test (LBIST) is used to detect latent faults at boot time. LBIST is enabled by an eFUSE and should only occur during the boot phase. To prevent inadvertent activation of LBIST during operation, the circuits to enable the test are implemented with DMR. The test will only be invoked when both the nominal and redundant circuits agree.

Clock Monitors

Xilinx SM tracking tags: [SM_OCMADMA_LOG1] [SM_PMU_LOG8] [SM_CSU_LOG8] [SM_LPD_LOG1] [SM_CRL_LOG2]

There are eight Clock Monitors (CLKMON) provided in the LPD which should be configured to verify the frequency of various clocks. Each CLKMON can be configured to select one of eight clocks to be monitored by setting the clka_mux_ctrl bits in the corresponding CHKRx_CTRL register (where 'x' is the number of the CLKMON in the range 0 to 7). The selected clock will be valid when the CLKMON measurement during the time relating to the value defined in the CHKRx_CLKB_CNT value falls within lower and upper threshold values defined in the CHKRx_CLKA_LOWER and CHKRx_CLKA_UPPER registers.

The clocks that should be monitored to increase fault coverage are as follows.

- cpu_r5_clk - clock used by the RPU
- ocm_r5_clk - clock used by the OCM
- lpd_switch_clk - clock used by interconnect
- lpd_lsbus_clk. - clock used by the low speed bus interconnect
- pmu_clk - clock used by the PMU.
- csu_pll_clk. - clock used by the CSU

Note that the PMU and CSU are both clocked by an internal ring oscillator (IRO) with a relatively large potential frequency range for valid operation (e.g., 146 MHz to 220 MHz). Although the same clock is used, the CLKMONs are sampling the clock from different nodes physically located within the regions occupied by the PMU and CSU to make the measurements, thereby verifying each connection to the source separately.

When a CLKMON detects that a clock is outside of the defined range or the CLKMON measurement has resulted in an internal counter overflow then the corresponding status bit of the 16 contained in the CLKMON_STATUS register will be asserted. The CLKMON_STATUS register can be inspected at regular intervals or the CLKMON_MASK and ERROR_EN_1 registers can be configured to propagate a CLK_MON error through to the ERROR_STATUS_1 register. ERROR_INT_MASK_1, ERROR_SIG_MASK_1, ERROR_POR_MASK_1, and ERROR_SRST_MASK_1 can be configured to further propagate or process the error as desired. The CLKMON_MASK register can also be configured generate Interrupt 60 to the RPU and APU.

Watchdog Timers

Xilinx SM tracking tags: [SM_RPU_WDT] [SM_PMU_WDT] [SM_CSU_WDT] [SM_CRL_WDT] [SM_OCMADMA_WDT] [SM_LPD_WDT] [SM_IOU_WDT]

Multiple watchdog timers are provided in the Low Power Domain (LPD) and should be used to detect and recover from system malfunctions. In normal operation, a CPU restarts the watchdog at regular intervals before the timer counts down to zero. If the timer does reach zero and the watchdog is enabled, then depending on the watchdog timer and its configuration a system reset or interrupt is generated or an external pin or SLVERR signal is asserted.

The PMUCSU watchdog is automatically enabled following a power-on reset and counts down from its maximum (24-bit) value. The purpose of this watchdog is to prevent system lockup even if the software becomes trapped in a deadlock during the boot phase. A timeout will lead to a POR, assertion of the EMIOWDTRSTO MIO pin and Interrupt 85.

The Low Power Domain System Watchdog Timer (LPD SWDT) consists of two triple-timer counters (TTCs). The timeout behavior is defined by the XWDTPS_ZMR_OFFSET register with the XWDTPS_ZMR_WDEN_MASK being the watchdog enable and then XWDTPS_ZMR_IRQEN_MASK and XWDTPS_ZMR_RSTEN_MASK bits defining the generation an interrupt or reset. The ERROR_EN_1 register can be configured to propagate a timeout through the LPD SWDT bit in the ERROR_STATUS_1 register. ERROR_INT_MASK_1, ERROR_SIG_MASK_1, ERROR_POR_MASK_1, and ERROR_SRST_MASK_1 can be configured to further propagate or process the error as desired. The LPD_SWDT is Interrupt 84 to the RPU and APU.

The Cortex-R5 MPCore forming the RPU has a dedicated system watchdog timer (RPU SWDT) and a dual triple-timer counter. This watchdog is designed to detect both systematic and random failures causing program flow errors

An AXI timeout block (ATB) acts as a watchdog timer for interconnect detecting hung communications through switches and at peripheral interfaces. The ATB_RESP_EN register is used to enable an ATB response. The ATB Err LPD is Interrupt 84 to the RPU and APU. The ATB_RESP_TYPE register is used to define if a local SLVERR is generated.

The OCM has its own watchdog timer. The forceeventforerrintsts and ForceEventforAUTOCMDErrorStatus registers are used to enable an interrupt.

Memory Built-In Self Test (MBIST)

Xilinx SM tracking tags: [SM_MBIST]

The Memory Built-In Self-Test (MBIST) is performed automatically by the PMU after POR and before release of reset to the CSU. MBIST is actually comprised of a set of MBIST controllers within the Processing System (PS). Between them, these controllers test all memories for latent faults. Coverage includes memories within peripherals and RPU as well as the OCM. Completion of each MBIST controller test is signified by one of the 32 status bits contained in the MBIST_DONE register with the result of each test reported by the corresponding 32 status bits contained in the MBIST_GOOD register.

Logic Built-In Self Test (LBIST)

Xilinx SM tracking tags: [SM_RPU_LBIST] [SM_PMU_LBIST] [SM_CSU_LBIST] [SM_OCMADMA_LBIST] [SM_INT_LPD_LBIST] [SM_CRL_LBIST]

The Logic Built-In Self-Test (LBIST) is used to detect latent faults at boot time when enabled to do so by an eFUSE. These automated tests focus on verifying the integrity of the fundamental control circuits and connectivity associated with the clocks and reset logic (CRL), RPU, PMU, CSU, OCM, LPD, DMA, and interconnect (e.g., bus switches). The PS_ERROR_STATUS pin will be asserted if any logic fails its test and the PS will not boot.

ROM Validation

Xilinx SM tracking tags: [SM_PMU_LOG9] [SM_CSU_LOG9] [SM_CRL_LOG3]

The CSU and PMU processors each have a ROM containing the code initially executed during the boot phase and with potential for further use during operation. As well as invoking the ROM validation prior to booting the device, the control circuits include measures to minimize the potential for unintentional activation of the ROM validation process during the boot and operational phases. The contents of each ROM are defined by the masks used when fabricating the silicon during the manufacturing process. ROM validation circuits automatically check the validity of the ROM contents after POR and before release of reset to the CSU. The validation process uses the SHA-3/238 engine to compute a hash which is compared with the expected hash value thereby verifying the SHA-3/238 engine at the same time. An incorrect hash value results in the PS_ERROR_OUT pin being asserted and the CSU being held in reset, aborting the boot process.

eFUSE and eFuse Cache Validation

Xilinx SM tracking tags: [SM_CSU_LOG11]

Each word stored in eFUSE memory is accompanied by a parity bit. Following the release of POR, the contents of the eFUSE memory is copied into a cache before being used by the CSU to implement a secure boot. If a parity error is detected during the copying process the CSU asserts the BR_ERROR status bit and sets error code 0x14 in the ERR_TYPE field of the CSU_BR_ERROR register. The default settings of the ERROR_EN_2 and ERROR_SIG_MASK_2 registers propagate this error through the PMU_PB bit of the ERROR_STATUS_2 register and assert the PS_ERROR_OUT pin.

The eFUSE cache can also be reloaded during operation using the EFUSE_CACHE_LOAD register. If a parity error is detected during the copying process the CACHE_ERROR status bit will be asserted in the EFUSE_ISR register. The EFUSE_IMR register can also be configured to generate Interrupt 119 to the RPU and APU.

PLL Lock Monitoring

Xilinx SM tracking tags: [SM_PLL_LOG1]

The lock status of the five PLLs within the LPD can be monitored by inspecting the VPLL, DPLL, and APLL status bits in the PLL_STATUS (CRF_APB) register and the RPLL and IOPLL status bits in the PLL_STATUS (CRL_APB) register. The ERROR_EN_2 register can be configured to propagate any of these PLL lock status bits through to the five PLL_LOCK bits in the ERROR_STATUS_2 register. ERROR_INT_MASK_2, ERROR_SIG_MASK_2, ERROR_POR_MASK_2, and ERROR_SRST_MASK_2 can be configured to further propagate or process the error as desired.

Isolation Self-Check

Xilinx SM tracking tags: [SM_INT_LPD_LOG2]

The AXI Isolation Block (AIB) is responsible for functionally isolating an AXI/APB master from a slave in preparation for an AXI/APB. The control circuits within the AIB include measures to minimize unintentional activation of any interface. Furthermore, these control circuits confirm when isolation is active and check that this is consistent with the current isolation request thus automatically canceling an unintentional activation should it occur.

Address Decode Error Detection

Xilinx SM tracking tags: [SM_PMU_LOG10] [SM_CSU_LOG10]

Physical resources are allocated known addresses or address ranges in the memory map. Additional address decoders have been implemented to detect when a master generates an address outside the used ranges. These circuits also detect hardware faults if an otherwise valid addresses issued by a master is corrupted en route.

If an attempt is made to access an address within the PMU or CSU that does not correspond with a physical resource then this otherwise un-decoded location will be detected. This error is first reported by the Status bit in the ADDR_ERROR_STATUS register. This is treated as a fatal error resulting in the automatic assertion of a reset to the PMU. The default settings of the ERROR_EN_2 and ERROR_SIG_MASK_2 registers further propagate this error through the PMU_UC bit of the ERROR_STATUS_2 register and assert the PS_ERROR_OUT pin.

Assumptions of Use of Non-LPD Modules

As well as the LPD, the user might also want to implement subsystems of the application software in the FPD and PL. In that case the item safety concept related to the user application shall define and implement the technical safety requirements allocated to those subsystems.

The next chapter Chapter 6, Zynq UltraScale+ MPSoC Subsystems Isolation, describes how the Zynq UltraScale+ MPSoC isolation features can be configured to guarantee the freedom from interference among the three modules, thus allowing ASIL decomposition or HFT=1.

Zynq UltraScale+ MPSoC Subsystems Isolation

Isolation Among Zynq UltraScale+ MPSoC Power Domains

The Zynq Ultrascale+ MPSoC has three main power domains, PS FPD, PS LPD, and PL. By design these power domains are separated from each other using a level shifter based gasket that supports clamp values on signals sourcing from an isolated domain. Each domains' isolation can be controlled from the PS PMU and the isolation can be customized to meet the system runtime requirements.

All system controls reside within the LPD power domain. Failure of this domain should be considered a major fault. Failure in the LPD power domain might cause a failure in the FPD due to the fact that both power domains make up the PS and share some resources sourced from the LPD. The PL, using the PS Only Reset mechanism (see Chapter 36 in UG1085 [Ref 4]), can be implemented in a way that it is more immune to failures in the FPD and LPD power domains.

Power Supply

The power supplies are designed as three external power rails that must be supplied by the user. All three power domains inside the device are presumed to be independent and are isolated from each other using a level shifter gasket. Failure of the LPD power supply will probably cause a fault in the FPD since the POR reset and REF_CLK for the FPD are sourced from the LPD. The PL power domain has fewer dependencies on the LPD power domain, assuming the PROG gate is used.

Clock (Separate PLL in FPD and LPD)

PS_REF_CLK is sourced into the LPD and sent to the FPD via the FPD-LPD level shifter gasket. PS_REF_CLK will continue to propagate to the FPD, even when functional isolation is applied to the other FPD-LPD interfaces. The FPD can also be setup to source a clock from the PL. The PL should be designed to source its clock from an independent source when design for safety use cases.

Reset Activation

The main system reset, PS_POR_B, enters into the LPD power domain. Failure on this reset cannot be isolated from the FPD. Using the PROG gate, the PL can be isolated from reset control in the LPD.

Test Mode Activation (Scan)

Inadvertent activation of test mode is unlikely as it requires both the Zynq UltraScale+ MPSoC to be booted in a test mode and for JTAG to then be used to request a scan. Alternatively, the picture debug register could be set in the clock-reset LPD controller to request scan mode. If desired, entry into the test mode can be blocked by setting the *Scan_Enable* bit of the *SAFETY_GATE (PMU_GLOBAL)* register to `0`.

Debug Activation

There are debug/test interfaces connected to the PL that are intended for use by Xilinx during the manufacture and testing of Zynq UltraScale+ MPSoC devices. Specific measures have been taken to ensure that these interfaces can have no effect on the PS during normal operation. There is no exposure to this interface within the Vivado design tools. The test pins are on a separate gasket that should remain isolated. In the event that this gasket should become un-isolated, the unconnected signals in the PS will be pulled high which is their non-active state.

Over Temperature

The system monitor located in the LPD (PS-SYSMON) is fully functional even when isolated. The FPD temperature sensor is not isolated. The response to over temperature in the FPD could be restricted based on the level of isolation. The system monitor located in the programmable logic (PL-SYSMON) can also be used (e.g., to monitor PL temperature) and can be accessed in two ways; via the direct APB connection to PL-SYSMON or via a user defined communication path implemented in the PL design. In both cases the LPD-FPD-PL isolation applies.

LPD–FPD–PL Isolation

Table 6-1 lists the different combinations of isolation that are possible by means of register REQ_ISO_TRIG bit fields.

Table 6-1: **REQ_ISO_TRIG, Bit-Field Details**

Field Name	Bit No.	Type	Reset value	Description
FP_LOCKED	4	Write-only	`0x0`	Locked Isolation Request Trigger for FPD
PL_NonPCAP	2	Write-only	`0x0`	Isolation Request Trigger for PL Interface not associated with PCAP

Table 6-1: **REQ_ISO_TRIG, Bit-Field Details *(Cont'd)***

Field Name	Bit No.	Type	Reset value	Description
PL	1	Write-only	0x0	Isolation Request Trigger for PL
FP	0	Write-only	0x0	Isolation Request Trigger for FPD

The assertion of bits #4 and #0 shall be mutually exclusive. The assertion of bits #2 and #1 shall be mutually exclusive. This makes some of the possible combinations *not allowed*, as identified in Table 6-2.

Table 6-2: **Possible Isolation Combinations Among LPD, FPD, PL**

Settings[1]					PMU_GLOBAL.PWR STATE[2]		States				
No.	FPD_LOCKED	PL_NonPCAP	PL	FPD	PL	FPD	LPD to FPD	LPD to PL	FPD to PL	Comments	Sequence
0[3]	0	0	0	0	1	1	ACTIVE	ACTIVE	ACTIVE	All domains are functional and can communicate.	Out-of-Reset state (see entry #4) + PL configured + PL Power Up Request Completed
1	0	0	0	1[4]	1	0	ISOLATED	ACTIVE	ISOLATED	FPD is non-functional and isolated from the system, LPD and PL can communicate. PL Must be configured.	Out-of-Reset state + PL configured + PL Power Up + FPD Isolation request
2	0	0	1	0	0	1	ACTIVE	ISOLATED	ISOLATED	LPD and FPD can communicate, PL Isolated from the system (PCAP is inactive)	Out-of-Reset state + PL Isolation request
3	0	0	1	1	0	0	ISOLATED	ISOLATED	ISOLATED	FPD is non-functional and isolated from the system, PL and LP are independent systems (PCAP is inactive)	Out-of-Reset state + PL Isolation request + FPD Isolation Request
4	0	1[5]	0	0	1	1	ACTIVE	CONFIG	ISOLATED	LPD-FPD Active, LPD can configure the PL	Out-of-Reset state[6]
5	0	1	0	1	1	0	ISOLATED	CONFIG	ISOLATED	FPD is non-functional, LPD can configure the PL	Out-of-Reset state + FP Isolation request
6	0	1	1	0	-	-	-	-	-	NOT ALLOWED	
7	0	1	1	1	-	-	-	-	-	NOT ALLOWED	
8	1	0	0	0	1	1	ISOLATED	ACTIVE	ACTIVE	FPD remains functional but isolated from LPD	
9	1	0	0	1	-	-	-	-	-	NOT ALLOWED	
10	1	0	1	0	0	1	ISOLATED	ISOLATED	ISOLATED	FPD remains functional and active to the PL, LPD is active but isolated from the system	
11	1	0	1	1	-	-	-	-	-	NOT ALLOWED	
12	1	1	0	0	1	1	ISOLATED	CONFIG	ISOLATED	FPD remains functional and LPD can configure the PL, FPD and PL can communicate	
13	1	1	0	1	-	-	-	-	-	NOT ALLOWED	
14	1	1	1	0	-	-	-	-	-	NOT ALLOWED	

Table 6-2: **Possible Isolation Combinations Among LPD, FPD, PL *(Cont'd)***

Settings[1]					PMU_ GLOBAL. PWR_ STATE[2]		States				
No.	FPD_LOCKED	PL_NonPCAP	PL	FPD	PL	FPD	LPD to FPD	LPD to PL	FPD to PL	Comments	Sequence
15	1	1	1	1	-	-	-	-	-	NOT ALLOWED	

Notes:
1. 0 => Interface with the other subsystems is active; 1 => Interface with the other subsystem is disabled (Isolated)
2. Read-only bits: 0 => Power Off; 1 => Power On.
3. The PL_NonPCAP interface can only be active once the PL is configured (pcfg_gwe=1 in the pcap_status register).
4. Anytime the FP (bit 0) is placed into full isolation, the FP will become unavailable. Functionally it will have its ref clock gated and will be clamped into reset.
5. To remove isolation a Power Up request shall be asserted.
6. In this context "Out-of-Reset state" means the state of the device once out of Reset.

As the Zynq UltraScale+ MPSoC comes out of reset it has the default configuration of No.4. In order to reach configuration No.0, after the PL has been configured a PL Power Up Request shall be issued.

Configurations No.8, No.10, and No.12 have all the LPD-FPD signal clamped except for controls, clock and reset signals Configurations No.1, No.3, and No.5 also have the controls, clock and reset signals clamped and the FPD is under reset.

Isolation with Arm TrustZone

ARM® TrustZone® technology is a system-wide approach to security. It is tightly integrated into the Cortex®-A53 processors and extends throughout the system via the AMBA® AXI™ bus to TrustZone enabled IP blocks. The TrustZone hardware architecture aims to provide a framework that allows the confidentiality and integrity of assets to be protected from specific attacks. Beside the mitigation of security threats, TrustZone characteristics can be used in a safety design as a decoupling layer providing freedom from interference between safety and non-safety application or between channels of safety applications.

TrustZone enables a single physical processor core to execute code safely and efficiently from both the Normal world and the Secure world (see Figure 6-1). The two virtual processors context switch via a new processor mode called monitor mode when changing the currently running virtual processor.

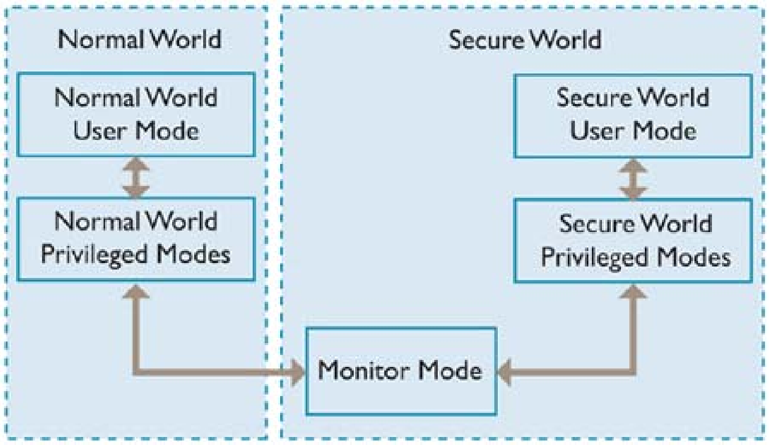

Figure 6-1: **ARM Trustzone Technology**

The mechanisms by which the physical processor can enter monitor mode from the Normal world are tightly controlled, and are all viewed as exceptions to the monitor mode software. Software executing a dedicated instruction can trigger entry to monitor, the Secure Monitor Call (SMC) instruction, or by a subset of the hardware exception mechanisms.

The software that executes within monitor mode is implementation defined. Access of Zynq programmable logic can be set as part of TrustZone allowing the implementation of monitors and additional diagnostic that assists the safety functionality in TrustZone.

Isolation with Xilinx Isolation Design Flow

The Isolation Design Flow (IDF) is a design methodology that ensures fault propagation can be controlled within the Zynq UltraScale+ MPSoC's programmable logic. It is a critical technology in achieving freedom from interference (FFI) and eliminating common cause failures (CCF). Originally pioneered for government cryptographic systems, it is now an integral part of the Xilinx IEC61508 certified tool chain.

IDF is a methodology utilizing advanced floor-planning techniques to physically isolate functionality in the PL. Functionality is isolated by a small *fence* of unused logic and programmable interconnect. The logic and routing of the isolated function is wholly contained within the isolated region. Xilinx invests significant resources into the analysis of the fence, ensuring the isolation required is indeed provided. Once the design is complete, verification of work products is done by a separate and independent tool called the Vivado Isolation Verifier (VIV). A high level design flow using IDF is shown in Figure 6-2.

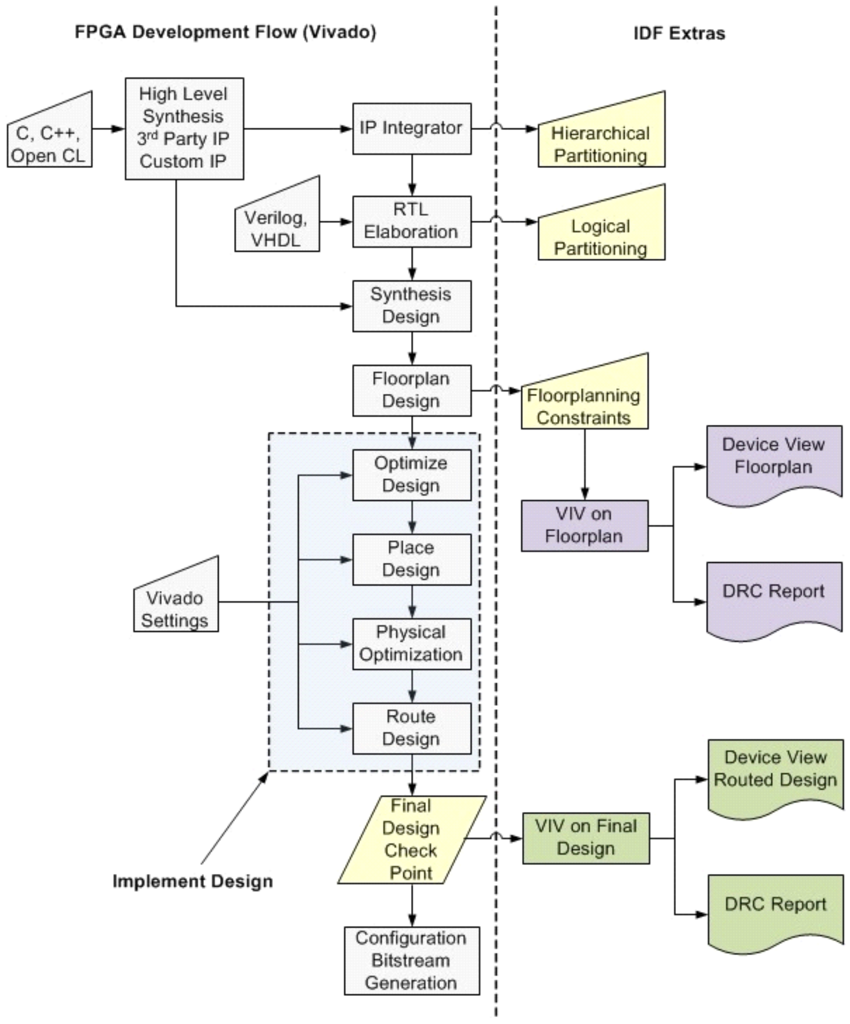

Figure 6-2: **Design Methodology using IDF**

A block diagram of a sample safety critical application requiring isolation, and the resulting implementation is shown in Figure 6-3.

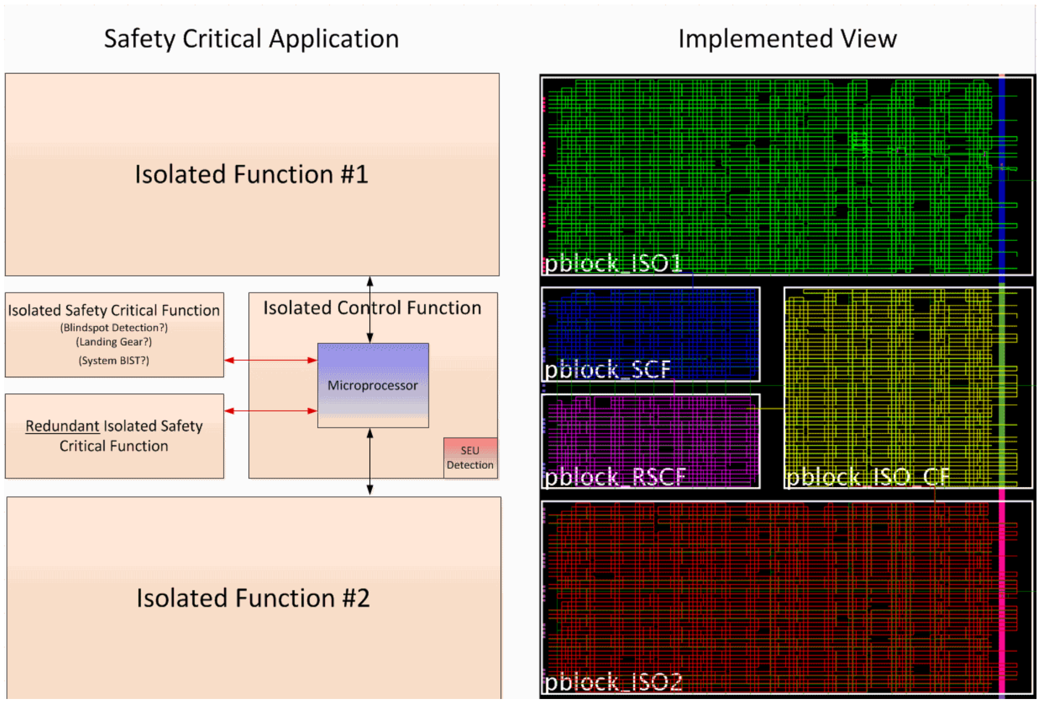

Figure 6-3: **Sample Design Isolated Using IDF**

The following detailed documentation is provided to aid a customer in the use and implementation of IDF:

- *Isolation Design Flow for Xilinx 7Series FPGAs or Zynq-7000 AP SoCs (Vivado Tools)* (XAPP1222) [Ref 11]
- *Zynq-7000 AP SoC Isolation Design Flow Lab (Vivado Design Suite 2015.1)* (XAPP1256) [Ref 12]
- IDF website: http://www.xilinx.com/applications/isolation-design-flow.html

Safety Mechanisms for Peripherals Protection

In general, peripherals are addressed by a mix of application level measures (i.e. by means of *assumptions of use*) and hooks at the HW level to simplify those application level measures.

Those measures are:

- Use of spatial redundancy (i.e. using two channels of the same peripheral and compare results)
- Use of time redundancy (e.g. repeating a certain operation twice and compare results)
- Use of application loop (e.g. detecting a fault in an output by reading back it and evaluate the correctness by SW)
- Use of loopback (e.g. CAN transmitter looped back to CAN receiver)
- Use of End-to-End (E/E) safety protocol

To give a concrete example, consider the CAN interface in LPD. The CAN protocol engine could be supervised by a combination of three main safety mechanisms:

- The possibility of performing a loop-back test
- The CRC included by default in the CAN basic data frame
- An additional CRC to be added on top of the standard CAN data frame to take into account the requirements of functional safety for off-chip data communication

In fact, both IEC 61508 and ISO 26262 require to cover at least the following failure modes:

- repetition of information
- loss of information
- delay of information
- insertion of information
- masquerade or wrong addressing of information
- incorrect sequence of information
- corruption of information

- asymmetric information sent from a sender to multiple receivers
- information from a sender received by only a subset of the receivers
- blocking access to a communication channel

Those requirements are typically fulfilled by adopting an end-to-end (E/E) safety protocol composed by the following functions. One or more of these functions may already be present when implementing a communication standard:

- a header or delimiter
- adding an additional CRC in the data field of the data frame (see Figure A-1)
- adding a frame counter inside the data field
- performing a timing check (window based) on the arrived packet.

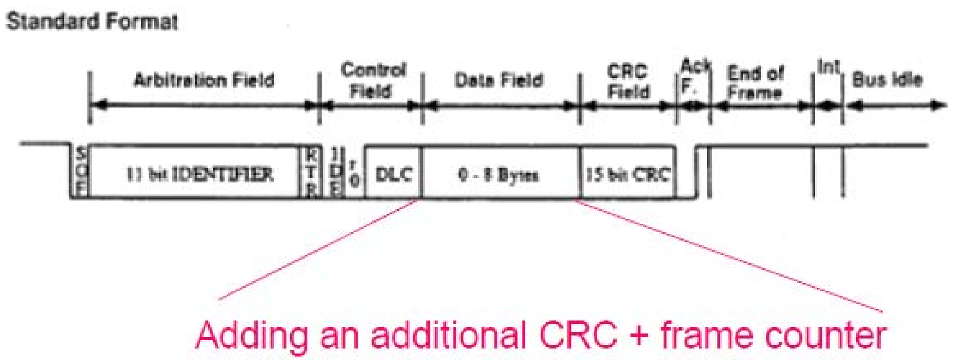

Figure A-1: **Extending the Can Data Field to Cover Communication Failures**

The E/E safety protocol will also cover other digital logic like the prescaler, the CAN mailbox (RAM), etc. as depicted in Figure A-2.

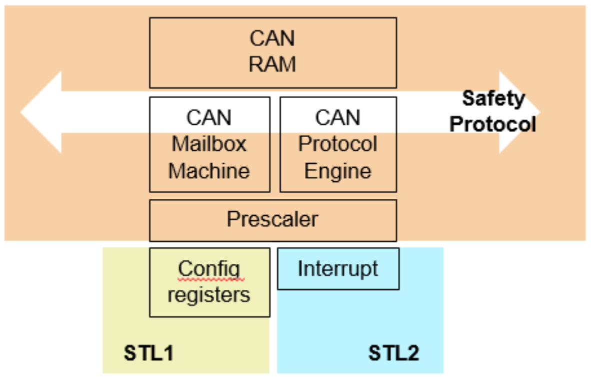

Figure A-2: **Summary of Safety Mechanisms for CAN**

In general, peripheral memories (such as CAN and RAM) are covered by the E/E safety protocol except for the portion storing configuration data, covered by a dedicated STL.

A similar approach can be implemented for all of the other safety-elated peripherals. For example, a simple way to cover faults in PWM in combination with the application is to implement a loopback test by means of a GPIO, as shown in Figure A-3. This scheme can also be used for other peripherals and of course for GPIO themselves.

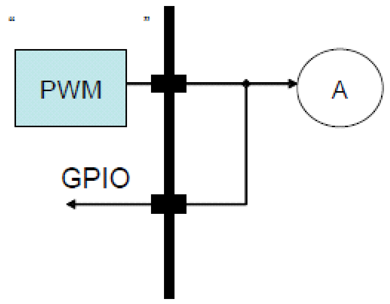

Figure A-3: **Testing the PWM with a GPIO Loopback Test**

The E/E safety protocol established on top of the communication interface, as shown Figure A-3, is also able to cover part of the interconnect between the peripheral to the respective CPU.

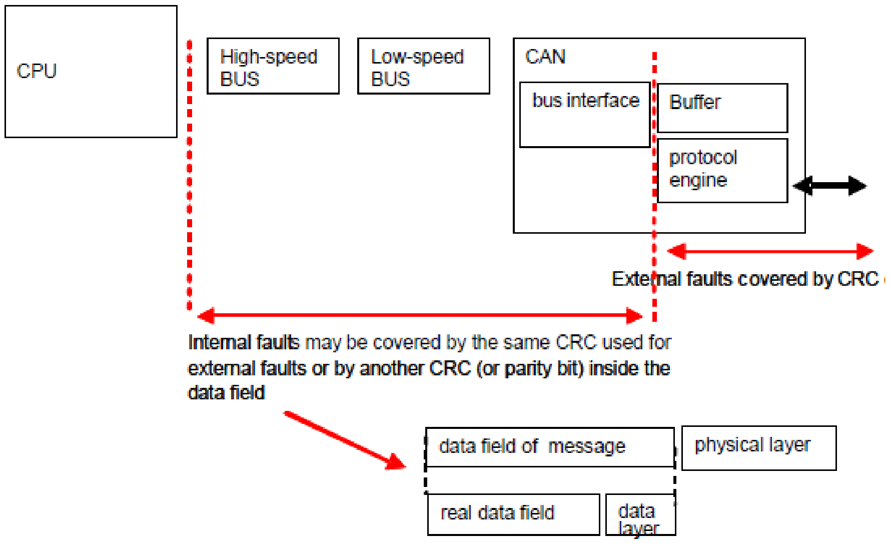
CPU
High-speed BUS
Low-speed BUS
CAN
bus interface
Buffer
protocol engine
External faults covered by CRC
Internal faults may be covered by the same CRC used for external faults or by another CRC (or parity bit) inside the data field
data field of message
physical layer
real data field
data layer

Figure A-4: **Covering Part of Bus through the E/E Safety Protocol**

List of Software Safety Mechanisms and Assumptions Of Use

Table B-1 reports the following information:

- Name of STL or AoU: name of software test library or assumption of use. This is the tag used in MPSoC safety analysis (done by Xilinx) and is reported as a label here.
  - Recommendation:
    - ++: implementation is highly recommended. If not implemented it would impact the MPSoC safety metrics (>2% for SPFM, or >10% for the latent metric, or > 1 FIT for PMHF)
    - +: implementation is recommended: If not implemented it would slightly impact the MPSoC safety metrics
    - o: optional. If not implemented it should not impact the MPSoC safety metrics. This option is being used for some STL only, for which the user might have particular reason to implement in his application
- Fault coverage for Permanent/Transient Faults:
  - Yes/No: the SM or AoU is/is not effective to control Permanent/Transient Faults
  - NA: Not Applicable for this particular SM or AoU
- Active at Startup/Run-time
  - Yes: the SM or AoU shall be active during startup/run-time.
    - in case of a SM, the User shall configure the SM to be effective during startup/runtime
    - in case of an AoU, the User shall implement the AoU in the Application in order to be effective during startup/runtime
    - in case of an STL, the User shall implement the STL as part of the Application start-up procedure
- STL or AoU:
  - STL: the User is supposed to integrate the STL (provided by Xilinx) in the Software Application
  - AoU: the User is supposed to implement the assumption of use in the software Application

Table B-1: **Software Safety Mechanisms and Assumptions of Use**

Name of STL or AoU	Recommendation	Fault Coverage for		Shall be Active at		STL or AoU	Referenced Section
		Permanent Faults	Transient Faults	Startup[1]	Run-time[2]		
MRD_AOU001	NA	NA	NA	NA	NA	AoU	Safety Goal
MRD_AOU002	++	Yes	Yes	NA	NA	AoU	Fault Tolerant Time Interval (FTTI) – Process Safety Time (PST)
SC_AOU003	++	Yes	Yes	NA	Yes	AoU	Power Supplies
SC_AOU004	++	Yes	Yes	NA	Yes	AoU	XMPU Configuration XPPU Configuration
SC_AOU005	++	Yes	Yes	NA	Yes	AoU	Power Supplies
SC_AOU006	++	Yes	Yes	NA	Yes	AoU	Clock Faults Detection
SM_CRL_APPEXWDT	++	Yes	Yes	NA	Yes	AoU	Clock Faults Detection
SC_AOU007	++	Yes	Yes	NA	Yes	AoU	LPD-DMA Usage
SM_IOU_WDT	+	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SM_RPU_APP1	++	Yes	Yes	NA	Yes	AoU	RPU Verification of Interrupts
SM_RPU_APP2	++	Yes	Yes	NA	Yes	AoU	RPU Communication with GIC
SM_RPU_APP3	++	Yes	Yes	NA	Yes	AoU	RPU AXI Communication
SM_IOU_APP1	++	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SM_QSPI_APP1	++	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SM_QSPI_APP2	+	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SM_CAN_APP1	++	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SM_CAN_APP2	+	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SC_AOU008	++	Yes	Yes	NA	Yes	AoU	XMPU Configuration XPPU Configuration
SC_AOU009	++	Yes	Yes	NA	Yes	AoU	XMPU Configuration XPPU Configuration
SC_AOU010	++	Yes	NA	Yes	NA	AoU	XMPU Configuration XPPU Configuration

Table B-1: **Software Safety Mechanisms and Assumptions of Use *(Cont'd)***

Name of STL or AoU	Recommendation	Fault Coverage for		Shall be Active at		STL or AoU	Referenced Section
		Permanent Faults	Transient Faults	Startup[(1)]	Run-time[(2)]		
SC_AOU011	++	Yes	Yes	NA	Yes	AoU	XMPU Configuration XPPU Configuration
SC_AOU013	++	NA	Yes	NA	Yes	AoU	PMU Processor Usage
SC_AOU014	++	NA	Yes	NA	Yes	AoU	PMU Processor Usage
SC_AOU015	++	Yes	Yes	NA	Yes	AoU	Dynamic Power Management in LPD
SC_AOU016	++	Yes	Yes	NA	Yes	AoU	DAP to be Turned Off USB
SC_AOU017	++	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SC_ AOU018	++	Yes	No	Yes	NA	AoU	System Monitors
SM_AMS_APP1	++	Yes	No	Yes	No	AoU	System Monitors
SM_AMS_APP2	++	No	Yes	No	Yes	AoU	System Monitors
SM_AMS_APP3	++	No	Yes	No	Yes	AoU	System Monitors
SC_ AOU019	++	Yes	Yes	NA	Yes	AoU	Integrity of the Configuration Registers
SC_AOU020	++	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SC_AOU021	++	Yes	Yes	Yes	NA	AoU	Quad-SPI for Boot and Configuration
SC_AOU022	++	Yes	Yes	NA	Yes	AoU	Quad-SPI Controller Protection
SC_AOU023	++	Yes	Yes	NA	Yes	AoU	Non-safety Related Modules and Features in LPD
MRD_AOU024	++	NA	NA	Yes	NA	AoU	Operating States – Safe States
SC_AOU030	+	Yes	No	Yes	NA	AoU	BIST Permanently Enabled
DFA_AOU001	+	Yes	No	Yes	NA	AoU	BIST Permanently Enabled
SC_AOU031	++	NA	NA	NA	NA	AoU	MBIST Usage
SC_AOU040	+	Yes	Yes	NA	Yes	AoU	Triple-Timer Counters in LPD
SC_AOU041	++	Yes	Yes	NA	Yes	AoU	RPU WDT and PMU WDT Configuration in LPD
SC_AOU042	++	Yes	Yes	NA	Yes	AoU	Non-safety Interfaces
SC_AOU043	++	Yes	Yes	NA	Yes	AoU	RPU CPU Configuration
MRD_AOU057	++	NA	NA	Yes	NA	AoU	Operating States – Safe States
SC_AOU302	++	NA	NA	NA	NA	AoU	Fault Handling Procedures

Table B-1: **Software Safety Mechanisms and Assumptions of Use *(Cont'd)***

Name of STL or AoU	Recommendation	Fault Coverage for		Shall be Active at		STL or AoU	Referenced Section
		Permanent Faults	Transient Faults	Startup[1]	Run-time[2]		
SM_INT_LPD_APP	++	Yes	Yes	NA	Yes	AoU	Peripherals, Interconnect, and Communication Channels Protection through CRC Protocols
SM_INT_APP1	++	Yes	No	No	Yes	AoU	Dynamic Power Management in FPD and PL
SM_CRL_LOG4	++	Yes	Yes	NA	Yes	AoU	Clock Faults Detection
SC_AOU400	+	Yes	Yes	Yes	Yes	AoU	Clock Faults Detection
SC_AOU300	+	NA	NA	NA	NA	AoU	Assumptions on STL Integration by the Application
SC_AOU301	+	YES	YES	NA	YES	AoU	Assumptions on STL Integration by the Application
SC_AOU303	++	NA	NA	NA	Yes	AoU	LPD-DMA Usage
SM_STL_GIC	++	Yes	No	NA	Yes	STL	Software Safety Mechanisms (Software Test Libraries)
SM_STL_REG	++	Yes	No	NA	Yes	STL	Software Safety Mechanisms (Software Test Libraries)
SM_STL_INTC	++	Yes	No	NA	Yes	STL	Software Safety Mechanisms (Software Test Libraries)
SM_STL_CLK	++	Yes	No	NA	Yes	STL	Software Safety Mechanisms (Software Test Libraries)
SM_STL_ATB	++	Yes	No	NA	Yes	STL	Software Safety Mechanisms (Software Test Libraries)
SM_STL_MEM	o	Yes	Yes	NA	Yes	STL	Software Safety Mechanisms (Software Test Libraries)

Notes:

1. *Active at Startup* means that the Safety Mechanism has to be applied during system boot/startup phase. This typically applies for mechanisms effective for latent faults (e.g. memory BIST).
2. Safety Mechanism active during run-time can be continuous or periodic. *Continuous* means that the SM is active during device mission and works in parallel with the monitored function (e.g. power monitor). *Periodic* means that the SM is active during device mission and is active at least every FTTI. Both methods are effective for single point faults.

ISO 26262-10:2011, Clause 9.2.2

Step 4 - Integration of the SEooC into the item

During item development, the safety goals and the functional safety requirements are specified. The functional safety requirements of the item are matched with the functional safety requirements assumed for the SEooC to establish their validity.

In the case of an SEooC assumption mismatch, a change management activity beginning with impact analysis is conducted as written in ISO 26262-8:2011, Clause 8 (Change management). Potential outcomes include:

–the difference can be deemed to be acceptable with regard to the achievement of the safety goal, and no action is taken;

–the difference can be deemed to impact the achievement of the safety goal, and a change can be necessary to either the item definition or the functional safety concept;

–the difference can be deemed to impact the achievement of the safety goal, and a change is required to the SEooC component (including possibly a change of component).

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

For continual updates, add the Answer Record to your myAlerts.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

Functional safety standard information:

1. ISO26262 - ISO 26262: 1,2,3,4,5,6,7,8,9: 2011(E) / ISO 26262-10: 2012(E)
2. IEC61508 - IEC61508:1-7 © IEC:2010
3. TR 62380 © IEC:2004(E)

Zynq UltraScale MPSoC technical documentation:

4. *Zynq UltraScale MPSoC Technical Reference Manual* (UG1085)
5. *Zynq Ultrascale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)
6. *UltraScale Architecture System Monitor User Guide* (UG580)

Zynq UltraScale MPSoC Functional Safety Related Documentation:

7. *Device Reliability Report* (UG116)

8. *Zynq UltraScale+ MPSoC Software Safety User Guide* (UG1220)
9. *Xilinx Quality Manual* (QAP0002)
10. *Vivado Safety Manual* (UG1187)

Documentation supporting use and implementation of IDF:

11. *Isolation Design Flow for Xilinx 7Series FPGAs or Zynq-7000 AP SoCs (Vivado Tools)* (XAPP1222)
12. *Zynq-7000 AP SoC Isolation Design Flow Lab (Vivado Design Suite 2015.1)* (XAPP1256)
13. IDF website: http://www.xilinx.com/applications/isolation-design-flow.html
14. *OS and Libraries Document Collection* (UG643)

Please Read: Important Legal Notices