This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

EUROPEAN AVIATION SAFETY AGENCY
AGENCE EUROPÉENNE DE LA SÉCURITÉ AÉRIENNE
EUROPÄISCHE AGENTUR FÜR FLUGSICHERHEIT

Research Project EASA.2012/04

COTS-AEH – Use of complex COTS (Commercial-Off-The-Shelf) in airborne electronic hardware – failure mode and mitigation

easa.europa.eu

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Disclaimer

This study has been carried out for the European Aviation Safety Agency by an external organization and expresses the opinion of the organization undertaking the study. It is provided for information purposes only and the views expressed in the study have not been adopted, endorsed or in any way approved by the European Aviation Safety Agency. Consequently it should not be relied upon as a statement, as any form of warranty, representation, undertaking, contractual, or other commitment binding in law upon the European Aviation Safety Agency.

Ownership of all copyright and other intellectual property rights in this material including any documentation, data and technical information, remains vested to the European Aviation Safety Agency. All logo, copyrights, trademarks, and registered trademarks that may be contained within are the property of their respective owners.

Reproduction of this study, in whole or in part, is permitted under the condition that the full body of this Disclaimer remains clearly and visibly affixed at all times with such reproduced part.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

EASA.2012.C15
"COTS-AEH" Project.

COTS-AEH - USE OF COMPLEX COTS (COMMERCIAL-OFF-THE-SHELF) IN AIRBORNE ELECTRONIC HARDWARE – FAILURE MODE AND MITIGATION

THALES AVIONICS

Dossier ref. CCC/13/001303– Rev. 05

Authors: Philippe BIETH, Vincent BRINDEJONC

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

REVISIONS

Revision	Date	Effect on §	Description
00.1	13/02/2013	-	Creation of the document
00.2	01/03/2013	All	First draft delivery to EASA
00	11/03/2013	All	First Release of the COTS-AEH Report
01	09/04/2013	All	First Release with integration of EASA comments
02	31/05/2013	All	Second Release of the COTS-AEH Report
03	07/06/2013	All	Third release for Cologne meeting
04	20/09/2013	5, 6, 7, 8, 9, 10	Fourth release: final draft for comments
05	27/11/2013	3, 5, 6, 9, 10	Final version with comments taken into account

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

Table of Content

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

List of Figures

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

List of table

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Thales Disclaimer

In accordance with the provisions of the contract ref. EASA.2012.C15 entered into with the EASA for the performance of the COTS-AEH study, some background intellectual property right shall remain Thales Avionics' property and can be reused without constraint by Thales Avionics.

Such Backgrounds are the following:

- Methods, techniques and associated results contained in the present report to the reference [31], [69];
- methods, techniques and associated results about "Monitoring by an external independent item on the data path" contained in section 9.3.4.4;
- methods, techniques and associated results about "Way to conduct functional and worse case tests" contained in section 9.2.3.1;
- the concept and its illustration displayed in the Figure #96;
- methods, techniques and associated results about "Way to conduct endurance tests" contained in section 9.2.3.2.

The other documents and publications provided as reference in this document shall remain the property of their respective owners.

This disclaimer is in complement and prevails on the European Aviation Safety Agency disclaimer stated above.

Thales Avionics SAS
18 Av Marechal Juin,
BP49
92362 Meudon-la-Forêt
FRANCE

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Acknowledgements

There are a number of people without whom this report might not have been written and whom the authors are greatly thankful.

Our first acknowledgments are dedicated to EASA. Mr Kleine-Beek has always been present for management advices. Comments and technical discussions with Richard Canis and Stéphane Vaubourg have considerably improved the report.

Within Thales many people contributed to this achievement. We would first like to thank on one hand the initiators Marc Gatti, Patrice Toillon, Thierry Filloux and on the other hand Joël Bosson, Eric Parelon, Cédric Chevrel and Marc Fumey for strategic discussions.

On the technical aspects the report benefited of the deep, synthetic and pedagogical understanding of Patrick Dervin on digital electronics. Many of these pages could not have been written without him. Guy Berthon was always available for meaningful comments and support. Christian Pitot introduced the complementarity between tests and mitigations. Many pages on multicores come from passionate discussions with Anthony Roger, Xavier Jean and Hicham Agrou. A large amount of knowledge from Anthony is in particular present in the multicore state of the art. The PCIe chapters are based on the knowledge of Aurélien Asséo who wrote the section on PCIe intrinsic failure mitigation mechanisms. The MIL-STD-1553 chapters benefited from the knowledge of William Terroy. We thank Stéphane Bazin to have brought to our attention very interesting references, David Faura for general discussions, Charles Kremer and Jean-Philippe Borel for their support. We thank also Freescale represented by Mr Eric Bost for many complementary explanations.

The report set off many reactions and comments by our colleagues and we thank Philippe Juhasz, Tarik Aegerter, Jean-Daniel Chauvet, and Sébastien Tricot for their help to robustify this document.

Finally our thoughts go toward our families for their patience during these busy months.

Meudon-La-Forêt November, 22nd, 2013

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Executive Summary

The intent of this report is to provide a global methodology to tackle design errors of COTS in airborne safety critical usage of Design Assurance Level A.

The methodology is deployed in two major steps

1. Analysis of failure mode of the COTS both at black box and grey box level in order to list the failure modes induced by potential design errors;
2. Definition of test in order to improve the qualitative confidence level on COTS blocks reliability and definition of detection and mitigation mechanisms for the analysed failures.
   Where Reliability is the ability to perform a function for a given duration in given conditions.

In order to achieve this goal the project has selected and analyses some Complex and Highly complex COTS families.

Black box approach relies on COTS output failure modes and intrinsic detection / mitigation mechanisms. Grey box approach relies on an intermediate model between black box and non-accessible white box level. The grey box model is chosen in order to be free of property rights and heuristic. It allows detailing the COTS block behaviours and failures and reaching a level of detail at which the possible causes of design errors can be guessed.

The methodology tries also to limit its extension in abstraction to a logical level, avoiding the details of physical level when possible and sorting these details when they are required. At this logical level the method relies on a generic failure model with five families of failures that are mapped on each COTS type of interfaces at black box level and on COTS block interfaces at grey box level.

In order to mitigate design errors conjectured in the first part of the work, test and detection/mitigation mechanisms are proposed.
Tests (in particular randomized endurance tests) allow improving the confidence on the reliability of some COTS blocks with respect to design errors.

The test function is

- To determine a domain of controllable determinism for COTS usage.

The detection and mitigation mechanisms have then two functions:

- To guarantee that the COTS remains in the domain of controllable determinism;
- To guarantee a controllable functioning in the defined controllable determinism domain;

Detection / Mitigations mechanisms are of three types:

- COTS internal detection and mitigation mechanisms;
- COTS detection mechanisms associated to architectural mitigation mechanisms;
- Architectural detection and mitigation mechanisms.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

All this mechanisms should be tested during development with fault injection type tests and monitored in operation by Power-On Built-In-Test in order to prevent adverse effects of systematic or random faults on their availability and correct functioning.

The discussion on a completion criterion on COTS study has been raised. The report does not intent to conclude on this question.

The key points of the study outlined here before could nevertheless provide some hints to the system and equipment development teams, safety engineers and certification authorities in order to work safely and efficiently.

This Document Provided by the Abbott Aerospace
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

1 Background

1.1 Concern

Usage of complex and highly complex Commercial-On-The-Shelf (COTS) components become more and more predominant in airborne systems, in particular for systems supporting critical functions (i.e. functions of Design Assurance Level (DAL) A or B see ED-79A / ARP4574A [1]). This is mainly due to a need of a higher integration (SWaP: Size, Weight and Power) of avionics platforms, in conjunction with the high evolution rate of the available supporting technologies on the market.

This trend has to be considered in association with an increase of the intrinsic complexity of these components that have experienced a rapid evolution from different components architected by the item / system designer on a Printed Circuit Board (PCB) to a fully integrated chip, named System on Chip (SoC), architected by the chip designer (thanks to the reduction of technology process size in components).

A typical example of this evolution can be observed on Personal Computer type board as depicted on **Figure 1** that presents a model of progressive integration from a microprocessor to an integrated microcontroller. In this evolution, the Points of Observation and Points of Control have disappeared for testing or analysis purposes. Thus the control of the component features is more difficult to achieve.

Each Integration step (Is) leads to the disappearance of Observable points from the point of view of system/item designer and increase the internal complexity of the COTS as it implements new electronic functions.

- At initial Integration step (Is0) the microprocessor, modelled here by its Core[1], communicates with external devices distributed on the PCB. Every communication line is observable and controllable if needed. The observable points were:
    - Communication between the Core and the North bridge,
    - Communication between the North bridge and the Memory,
    - Communication between the North bridge and the South bridge,
    - Inputs/Outputs (I/O) of the North and South bridges.

These Points of Observation allow control of the communication by a third party that could be another microprocessor or one of the chips cited on **Figure 1**.

[1] One of the latest examples of this standalone processing core is the Freescale MPC7448.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

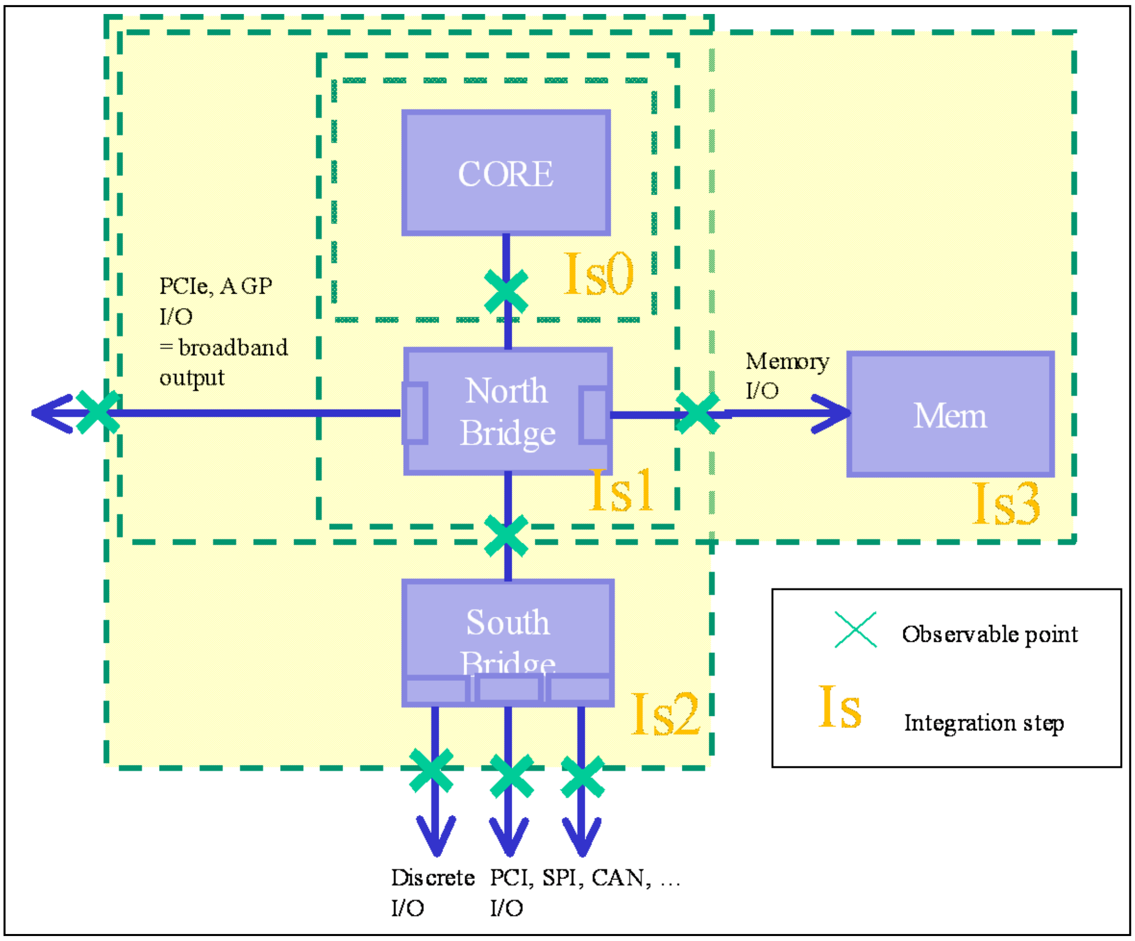


Figure 1: Typical evolution from microprocessor (IS0) to microcontroller (IS2 & IS3).

- At the first Integration step (Is1) all the communications between the Core and the North Bridge are internal and no more directly accessible to the system/item designer[2].

- Another Integration step (Is2) incorporates the South Bridge in the microcontroller perimeter, removing the observable point dedicated to the exchanges between the two bridges.

- Another Integration step (Is3) corresponds to the integration of Memory in the perimeter of the COTS. This leads to disappearance of the observable point between the Core and the Memory. It can be noted that some microcontrollers like TI-Hercule series embed directly Flash (up to 3MB) and RAM (up to 256kB) memory on the SoC.

  In fact, this integration began in the 90^{ies} with for instance the MC68HC11 from MOTOROLA. The novelty is now the development of microcontrollers with large computation capacities and integrated memories in particular different levels of Cache Memories that can be assimilated to internal memories.

[2] An example of such generation of microcontroller is the Freescale MPC8610 with a prominent North bridge allowing more I/O controlling.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

The lack of observation described here is not fully covered by the existing Guidelines such as ED-12()/DO-178() [2] [3] and ED-80/DO-254 [4]. In one hand, as mentioned in EASA CM - SWCEH – 001 [5] the microprocessor (to be understood here as the Core) is considered covered by ED-12()/DO-178() and then is out of scope of the report:

"The development assurance of microprocessors and of the core processing part of the microcontrollers and of highly complex COTS microcontrollers (Core Processing Unit) will be based on the application of ED-12B/DO-178B to the software they host, including testing of the software on the target microprocessor/microcontroller /highly complex COTS microcontroller." - EASA CM - SWCEH – 001- [5].

In the other hand, *ED-80/DO-254* [4] *recommends* for the COTS to obtain as many life cycle data as possible in order to gain in certification credit.

From the short overview above, we can see that COTS devices now available on the market to meet increased demand for increased functional characteristics and performance are moving towards the Highly Complex (HC) territory. New approaches to safety assurance might then be necessary as compared to detailed internal functional and dysfunctional analyses that were previously the preferred routes to support demonstration of acceptable behaviour. Moreover, the internal design of COTS is generally kept under proprietary rights by device manufacturers, hence precluding access to data potentially useful to reach the level of details similar to the one if it were designed by the avionics manufacturer. This is raising issues and concerns for avionics design where the concept of "intended functional performance under all foreseeable conditions with no anomalous behaviour" is one of the main guidelines. In addition, the production volumes in the aeronautics domain are limited compared to those of other industries as telecommunication, consumer electronics and information technology or automotive. This results in difficulties to find COTS suppliers ready to fully open their books.

1.2 Purpose of the Survey

Guidance have been provided via the EASA Certification Memorandum EASA CM - SWCEH – 001 [5] on the way to provide data and justifications for COTS. Section 9 of this EASA CM recommends activities to be performed in order to guarantee an acceptable level of confidence. These activities are as follows:

[1] Classification with respect to criticality (DAL) and complexity (SHE, CEH, HC),

[2] Device data from the COTS manufacturer (Data and Errata sheets, User's Manual, etc.),

[3] Design data, possibly additional data from the COTS manufacturer, subject to agreement,

[4] Usage Domain for the intended usage (e.g.: Used/Unused functions, usage conditions, etc.).

[5] Definition, and V. & V. of the Usage Domain (UD), including for Determinism and Partitioning,

[6] Errata sheets Capture and Control, and [7] Errata sheets Assessment and mitigations),

[8] Past and Current Experience gained along with recommendations (Errata workarounds),

[9] Configuration Management, including device data change information and description,

[10] Additional Verifications based on Change Impact Analysis (CIA) when device is changed,

[11] Validation & Verification versus requirements at upper level of hardware integration,

[12] Failure Analysis (modes and rates), including for configured used/unused functions,

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

[13] Product Service Experience (identification, documentation and assessment versus DAL),

[14] Evidence of Stability and Maturity of the COTS device, errata rate, and modifications,

[15] Architecture Mitigation and Common Cause (and Mode) Analysis for critical DAL A failure paths,

[16] Robust Partitioning (if used), and any other alternative methods (by design, tests, analyses).

When COTS contributes to Catastrophic event, , with a classification DAL A, the detection/ mitigation activities are recommended whatever this COTS is classified simple, complex or highly complex and whatever the Product Service Experience (PSE) is classified sufficient or low.

Such detection / mitigation activities have already been applied in avionics safety related applications even if the mitigation in place were principally to robustly deactivate some COTS feature. For instance, in the past, microcontroller I/O complex features have been used for maintenance purposes in ground life phases and as standalone micro-processors with simple I/O for in-flight safety related applications, with strong internal limitation mechanism in order to avoid any triggering of microcontrollers full features.

Another widely used, architectural mitigation is implemented by adding a control path – independent to the functional path in which the COTS component is involved. Sometimes it may also be acceptable to implement architectural mitigation through an independent means which could detect the COTS component failure. This last type of architectural mitigation is based on the assumption that any sudden internal failure of the COTS component can be identified when evaluating the output of the component - this is known as fault symptom detection. A fault symptom is defined as the consequence of the COTS internal failure (permanent or temporary) at the output of the component, e.g. frozen data, bits swap, timing shifts.

The present report provides few directions and ideas in how to complement, substantiate and justify the approaches already taken in the analyses of past and current designs involving Complex to Highly Complex COTS Electronic Hardware, in terms of detection and mitigation of COTS failures.

1.3 Methodology Outline

The call for tender of the project[3] proposes different tasks to be performed in order to complete the study. These tasks cover three main topics:

- COTS internal model and faults,
- COTS output flows and failures,
- Output flow failures detection and COTS faults mitigation.

During the development of the project it has been considered, by the authors, that this initially proposed, bottom-up methodology, was not sufficient to take into account the characteristics of COTS, in particular in their complexity and their property right aspects.

By the way, the methodology followed in this report has two viewpoints:

1. A top down approach starting from output flow characterizations and failures that provides,

[3] See "Specifications attached to the Invitation to Tender EASA.2012.OP.26 COTS-AEH — Use of complex COTS (Commercial-Off-The-Shelf) in airborne electronic hardware –failure mode and mitigation"

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

- A list of possible failures modes of the COTS as seen from the outside,
- A list of available internal protections depending upon I/O technology,
- Some possible COTS internal faults that can causes these failures.

These failures constitute an invariant for further studies on future COTS.

2. A bottom-up study on some typical architectures of complex COTS that provides
   - A list of possible faults of COTS internal blocks,
   - A list of COTS intrinsic typical mitigation mechanisms,
   - The effects of COTS internal faults on external failures (if any).

3. A proposal for detection means on the output flow and a corresponding mitigation of COTS faults.

1.4 Structure of the Report

This report is organized as follows. After this introductory chapter (Background), the aim and objectives agreed at the start of the study is recalled, and the changes occurred during the project are outlined (Chapter 2).

Then a complete glossary including acronym list and definitions is provided (chapter 3. Glossary). The two following chapters give a state of the art from literature point of view (chapter 4. Literature Review) and from technical point of view (chapter 5. state of the art).Afterward, the methodology followed is detailed (Chapter 6. Methodology) and the technical work deployed.

In chapter 7-dedicated to the study of interface failure modes - the top-down part of the study is detailed on the different output flow types, as they are partly independent of the COTS type. In this chapter, failure mitigation mechanisms previewed on the interface are clearly separated from those integrated into COTS (Chapter 8) and from mechanisms to be defined in order to mitigate COTS faults (chapter 9).

In chapter 8 each category of COTS is studied, with respect to their internal block faults. Due to the large number of possible scenario that could lead to detection, mitigation, Chapter 9 proposes the writing and test of a usage specification of the COTS that could cover many design errors. On the basis of chapters 7 and 8 detection and mitigation of remaining design failures can be conducted. This is detailed in chapter 9 before an overview and result summary in conclusion (chapter 10).

A dedicated chapter is reserved for references. Some details have been grouped in Annexes in order to avoid an overload of the report core.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

2 Aims and Objectives

This chapter is directly derived from the Specifications attached to Invitation to Tender EASA.2012.OP.26 and from EASA expectations expressed in the Terms Of Reference (TOR), as part of those specifications.

2.1 EASA Expectations

This research project is intended to investigate and identify under which conditions the assumption is valid that any sudden internal failure of the COTS component can be identified when evaluating the output of the component.

Therefore the objective of the study is to provide the Agency with sufficient data and analysis to be able to write and publish guidance material on the subject of COTS components fault symptoms and related generic detection means in safety-critical airborne systems. These systems would be of Development Assurance Levels (DAL) A in compliance with CS 25.1309 (a) and (b), ED-79A / ARP4574A, ED-80 / DO-254 and EASA Certification Memoranda (EASA CM-SWCEH–001 issue 01 Rev 01) for Airborne Electronic Hardware (AEH).

The scope of the study shall cover all kinds of complex and highly complex digital electronic hardware COTS components used in airborne safety critical applications (DAL A)

2.2 Project objectives

COTS-AEH project objectives are:

- To identify the typical complex COTS used and that will be used in future designs;
- To identify complex COTS-AEH typical failure modes;
- To perform investigations on the possibility to perceive complex COTS- AEH failure modes;
- To identify architecture detection/ mitigation means for complex COTS- AEH failure modes;
- To suggest recommendations for detection/ mitigation of complex COTS- AEH failures;
- To suggest recommendations and good practices;
- And to suggest complementary or amendments to EASA guidance.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

3 Glossary

3.1 ACRONYMS

AEH	**A**irborne **E**lectronic **H**ardware
BER	**B**it **E**rror **R**ate
CAT	**CAT**astrophic
COTS	**C**ommercial-**O**ff-**T**he-**S**helf
CPU	**C**entral **P**rocessing **U**nit
CRC	**C**yclic **R**edundancy **C**heck
CS	**Chip Select**
DAL	**D**evelopment **A**ssurance **L**evel
DDR	**D**ouble **D**ata **R**ate
DLL	**D**ata **L**ink **L**ayer
DMA	**D**irect **M**emory **A**ccess
DRAM	**D**ynamic **R**andom **A**ccess **M**emory
EASA	**E**uropean **A**viation **S**afety **A**gency
ECC	**E**rror **C**orrecting **C**ode
EEPROM	**E**lectrically-**E**rasable **P**rogrammable **R**ead-**O**nly **M**emory
FAA	**F**ederal **A**viation **A**dministration
FSB	**F**ront **S**ide **B**us
GPIO	**G**eneral **P**urpose **I**nput **O**utput
Gbps	**G**iga **b**its **p**er **s**econd
GBps	**G**iga **B**ytes **p**er **s**econd
I/O	**I**nput(s) / **O**utput(s)
I²C	**I**nter-**I**ntegrated **C**ircuit
ICD	**I**nterface **C**ontrol **D**ocument
IMA	**I**ntegrated **M**odular **A**vionics
IOMMU	**I**nput **O**utput **M**emory **M**anagement **U**nit
IP	**I**ntellectual **P**roperty
JEDEC	**J**oint **E**lectron **D**evices **E**ngineering **C**ouncil
LRU	**L**ine **R**eplaceable **U**nit
Kb	**K**ilo **b**it
KB	**K**ilo **B**yte
MB	**M**ega **B**yte
MBps	**M**ega **B**yte **p**er **s**econd
MBU	**M**ultiple **B**it **U**pset
MCU	**M**icro **C**ontroller **U**nit
MMU	**M**emory **M**anagement **U**nit
MPIC	**M**ulticore **P**rogrammable **I**nterrupt **C**ontroller
NDA	**N**on-**D**isclosure **A**greement
PCB	**P**rinted **C**ircuit **B**oard
PCI	**P**eripheral **C**omponent **I**nterconnect
PCIe	**P**eripheral **C**omponent **I**nterconnect **E**xpress
PIC	**P**rogrammable **I**nterrupt **C**ontroller

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

PLD	**P**rogrammable **L**ogical **D**evice
PLL	**P**hase-**L**ocked **L**oop
PoC	**P**oint **o**f **C**ontrol
PoO	**P**oint **o**f **O**bservation
RAM	**R**andom **A**ccess **M**emory
SDRAM	**S**ynchronous **D**ynamic **R**andom-**A**ccess **M**emory
SEU	**S**ingle **E**vent **U**pset
SoC	**S**ystem **O**n **C**hip
SPI	**S**erial **P**eripheral **I**nterface
SRIO	**S**erial **R**apid **IO**
TL	**T**ransaction **L**ayer
TLP	**T**ransaction **L**ayer **P**acket
TTL	**T**ransistor-**T**ransistor **L**ogic
UART	**U**niversal **A**synchronous **R**eceiver **T**ransmitter

3.2 KEY DEFINITIONS

3.2.1 Definitions relatives to hardware items classification

Commercial Off-The-Shelf (COTS) Component –
Component, integrated circuit, or subsystem developed by a supplier for multiple customers, whose design and configuration is controlled by the supplier's or an industry specification.
ED-80/DO-254 – Appendix C – Glossary of terms [4]

Complex hardware item
A hardware item is identified as simple only if a comprehensive combination of deterministic tests and analyses appropriate to the design assurance level can ensure correct functional performance under all foreseeable operating conditions with no anomalous behaviour. When an item cannot be classified as simple, it should be classified as complex.
ED-80/DO-254 §1.6 [4]

Complex COTS Microcontroller –
Any electronic item, which executes software in a specific core area and implements complex peripheral hardware elements such as I/O bus controller,
EASA CM – SWCEH – 001 [5]

Highly Complex COTS microcontroller
A microcontroller should be classified as Highly Complex as soon as it has any of the following characteristics:

- More than one Central Processing Unit (CPU) are embedded and they use the same bus (which is not strictly separated or which uses the same single port memory);
- Several controllers of complex peripherals are dependent on each other and exchange data;
- Several internal buses are integrated and are used in a dynamic way (for example, a dynamic bus switch matrix).

EASA CM – SWCEH – 001 [5]

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

Simple COTS Microcontroller –
Any electronic item, which executes software in a specific core area and implements simple peripheral hardware elements such as UART, A/D, D/A
EASA CM – SWCEH – 001 [5]

System on Chip (SoC) –
A System on Chip embeds in a single chip all the heterogeneous hardware functions necessary for a complete system. SoCs are usually made up of processor cores and other functions such as interface controllers, internal bus controllers, co-processors, on-chip memory, data converters…
The SoC components can be split into 2 main categories:
• The microprocessor based SoC
• The custom SoC (PLD/ASIC)
From Faubladier, F.; Rambaud, D 2008 [6]**,**

3.2.2 Definitions relative to Board and COTS Architectures

Address *(in this report)*
Part of informational content of a message necessary to its transmission[4]

Architecture *(in this report)*
Identification of a system's or an item's physical components and their interrelationships. For the particular purpose of this document, system or item has to be understood as the COTS and its direct peripherals within the computer.

Black Box
A device, system or object which can be viewed in terms of its input, output and transfer characteristics without any knowledge of its internal workings
(Wikipedia)

Cache
High-speed memory containing recently accessed data or instructions,

Control *(in this report)*
Part of informational content of a message necessary to its processing but is neither data nor destination address – see footnote [4] for an example -

[4] Consider the analogy with a letter;
- The envelope and its content constitute a message;
- This letter carry information in particular through the text written on the inserted paper but also, the address of the receiver, the address of the sender as it is written on the back side of the envelope or on the inside paper, the post-stamp identifying the sender post office and the sending date, …
- These information can be extracted of the analysis of
    - Payload included in the envelope (the letter);
    - Address on the envelope;
    - Encoded Controls on the envelope (stamp, post stamp).

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

Data *(in this report)*
Encoded syntactic content of the totality or of a part of a message[5]; (for instance in communication buses where data are separated from control and addresses)

Grey Box *(in this report)*
Black box refined breakdown level, built with fragmentary, non-contractual and potentially under NDA information

Information *(in this report)*[6]
Semantic content of a message through payload, addresses or controls – see footnote [4] for an example –

Intellectual Property (IP): In electronic devices, an Intellectual Property (IP) or Intellectual Property core (IP core) is an electronic function designed to be reused as a portion of a device (COTS, ASIC or PLD). [6]

Message *(in this report)*
A continuous block of data[7] with a defined length which is transported by the system (either by a communication network or within a module), -DO-297 [7]– see footnote [4] for an example -

North Bridge
The Northbridge has historically been one of the two chips in the core logic chipset on a PC motherboard. Dedicated to higher capabilities of the motherboard, it has increasingly migrated to the CPU chip itself. [8]

Payload *(in this report)*
Part of information carried by a message and related to end-user need – see footnote [4] for an example -.

South Bridge
The Southbridge is one of the two chips in the core logic chipset on a personal computer (PC) motherboard. The Southbridge typically implements the slower capabilities of the motherboard. [8]

Point of Control (PoC)
Physical location or connection accessible for injection of test signals or data;

Point of Observation (PoO)
Physical location or electrical connection accessible for measurement or logging of signals or data in reaction to injection of test signals or data at PoC

[5] Although we tried to give the most general definition of data, it can have different extents depending on the context, in order to cope with the various domains covered. For instance in communication buses context data is separated from the addresses and controls and represent the payload with possible particular controls (like those embedded on a letter), in memory context, address may be included in the range of data…

[6] This definition is freely adapted from Shannon, 1948 [72]. Contrary to Shannon, emphasis is placed here on semantic and not syntax.
[7] To be fully coherent with data definition, this definition should contain “data", "address" and "control”.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

3.2.3 Definitions relative to risk analysis

Failure
The inability of a system or system component to perform a required function within specified limits;
A failure may be produced when a fault is encountered.
ED-80/DO-254 [4]

Failure Mode:
The way in which the failure of an item occurs
ED-80/DO-254 [4]

Fault:
(1) A manifestation of a flaw in hardware due to an error or random event. A fault, if it occurs, may cause a failure. (2) An undesired anomaly in an item.
ED-80/DO-254 [4]

Error:
A mistake in requirements, design or implementation
ED-80/DO-254 [4]

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

4 Literature Review

4.1 Safety related avionic standards & guidelines

- ***ED-79A/ARP-4754A: Guidelines for Development of Civil Aircraft and Systems European Organisation for Civil Aviation Equipment (EuroCAE) & Society of Automotive Engineers (SAE), 2010.*** [1]
  This guideline addresses problematic that deal with complex embedded systems, included but not restricted to digital avionics systems. The definition of system in ED-79A/ARP-4754A is very wide and thus it can be applied to COTS.

- ***ED12()/DO-178()[8]: Software consideration in airborne systems and equipment certification.*** [2] [3]
  Although dedicated to software, this standard is referenced in EASA CM - SWCEH – 001 to reach development assurance for processors.

- ***ED-80/DO-254: Design Assurance Guidance for Airborne Electronic Hardware.*** [4]
  EURopean Organisation for Civil Aviation Equipment (EUROCAE and Radio Technical Commission for Aeronautics (RTCA).
  This standard deals with design quality for hardware elements. It explicitly addresses the COTS problematic.

- ***EASA CM - SWCEH – 001 Iss. 1 Rev. 1: Development Assurance of Airborne Electronic Hardware***, 9th Mar. 2012[9]. [5]
  This certification memorandum has been developed by EASA to highlight issues that shall be addressed in the certification process. It explicitly addresses the use of COTS for DAL A, B and C applications and considered the use of architectural mitigation:

[8] In this document reference to ED12()/DO-178(), with empty parenthesis, corresponds to both versions B and C of the standard.
[9] EASA CM - SWCEH – 001 Iss. 1 Rev. 1 can be download on the EASA web site : http://easa.europa.eu/certification/docs/certification-memorandum/EASA CM-SWCEH-001 Issue 01 Rev 01 Development Assurance of Airborne Electronic Hardware.pdf

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

9.3.8. Architectural mitigation
Results of the common mode analysis should be taken into account in order to show whether:
[15]: Architectural mitigation should be implemented in any case in which one or more instances of the COTS component could cause a Catastrophic failure effect without any other contributing faults occurring. The results of Common Cause Analysis performed by the applicant should be taken into account. For example, the anomalous behaviour or failure of identical COTS components (common design), implemented in redundant system architecture, should not lead to a Catastrophic failure condition.
Also the Common Cause Analysis performed at Aircraft level may reveal some Hazardous engine/propeller Failure Conditions that lead to a Catastrophic Aircraft Failure Condition. In such a case, this topic [15] should be addressed.

4.2 Others industrial sectors standard

ISO-26262, 2011 "Road vehicles — Functional safety" [9]

- **Part 4: Product development: system level**
- **Part 5: Product development: hardware level**
- **Part 10: Guideline on ISO 26262**

The automotive standard for functional safety introduces to particular concepts of interest for complex COTS architecture detection/ mitigation means. Firstly, it introduces the notion of safety context and in particular technical safety concept (ISO 26262 Part 4).

<u>Technical Safety Concept</u>
Specification of the technical safety requirements and their allocation to system elements for implementation by the system design;

The technical safety concept is a way to identify and specify the derived requirements associated to safety mechanisms in a safety-oriented design.

Secondly, this standard in its part 5 – Annex D (completed in the particular case of microcontrollers by part 10 Annex A) details the failure modes of the constitutive part of a COTS and the order of magnitude accessible for typical detection covering rates.

4.3 Studies on COTS–AEH usage in Avionics

- ***COTS CPU Selection Guidelines for Safety-Critical Applications*** *by* Forsberg, H. & Karlsson, K. [10]***;***
  Although it is quite old, this conference paper propose architectural detection/ mitigation in the case of COTS microprocessors avionics usage.

  - ***Handbook For The Selection And Evaluation Of Microprocessors For Airborne Systems*** *by Green, B. et al* [11]
    (<u>http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/AR_11_2.pdf)</u>
    This Handbook synthetize the results of a large study conducted for FAA between 2005 and 2011

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

by a team conducted by Pr. R. Mahapatra from Austin University on Microprocessors for avionics usages. These study led to several reports in 2006 [12], 2008 [13] [14], 2010 [15] and 2011 [16]. Starting from simple standalone microprocessors like Freescale PowerPC 7447, on which exhaustive models can be performed, authors identify the increasing difficulty to conduct such approach for complex microcontrollers like MPC8540 and finally favour Safety Net based approaches. Report and Handbook identify failure modes of microcontrollers and some detection/ mitigation techniques that can be applied.

- ***SoC Survey Report - Safety Implications of the use of system-on-chip (SoC) on commercial of-the-shelf (COTS) devices in airborne critical applications*** by Faubladier, F. & Rambaud, D on EASA – Research Project EASA.2008./1, ***2008***[10] [6]
  This EASA project report assesses the safety implication of using SoC in safety critical applications. It proposes amending recommendations for hardware certification process.

- ***The Use of Multicore processors in Airborne Systems by*** Jean, X., Gatti, M., Berthon, G., Fumey, M. for Research Project EASA EASA.2011.6, ***2011***[11] [17] offers a complete panorama of multicore possible usage in avionics with recommendation to use them safely.

[10] EASA – Research Project report EASA.2008/1 can be downloaded on the EASA web site: http://www.easa.europa.eu/safety-and-research/research-projects/docs/large-aeroplanes/Final_Report_EASA.2008_1.pdf.

[11] EASA – Research Project report EASA.2011/6 can be download on the EASA web site : http://www.easa.europa.eu/safety-and-research/research-projects/docs/large-aeroplanes/CCC_12_006898-REV07_-MULCORS Final Report.pdf

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

5 STATE OF THE ART

5.1 GENERIC EMBEDDED COMPUTING ARCHITECTURE

The present chapter aims at presenting the State of the Art on technologies of COTS handled in this report. Figure 2 presents a typical state-of-the-art computing architecture that identifies in their context, potential COTS candidates for assessment in the present study. Depending upon the kind of computing platform concerned Figure 2 can represent either a part of an electronic board, an electronic board or a complete computer.

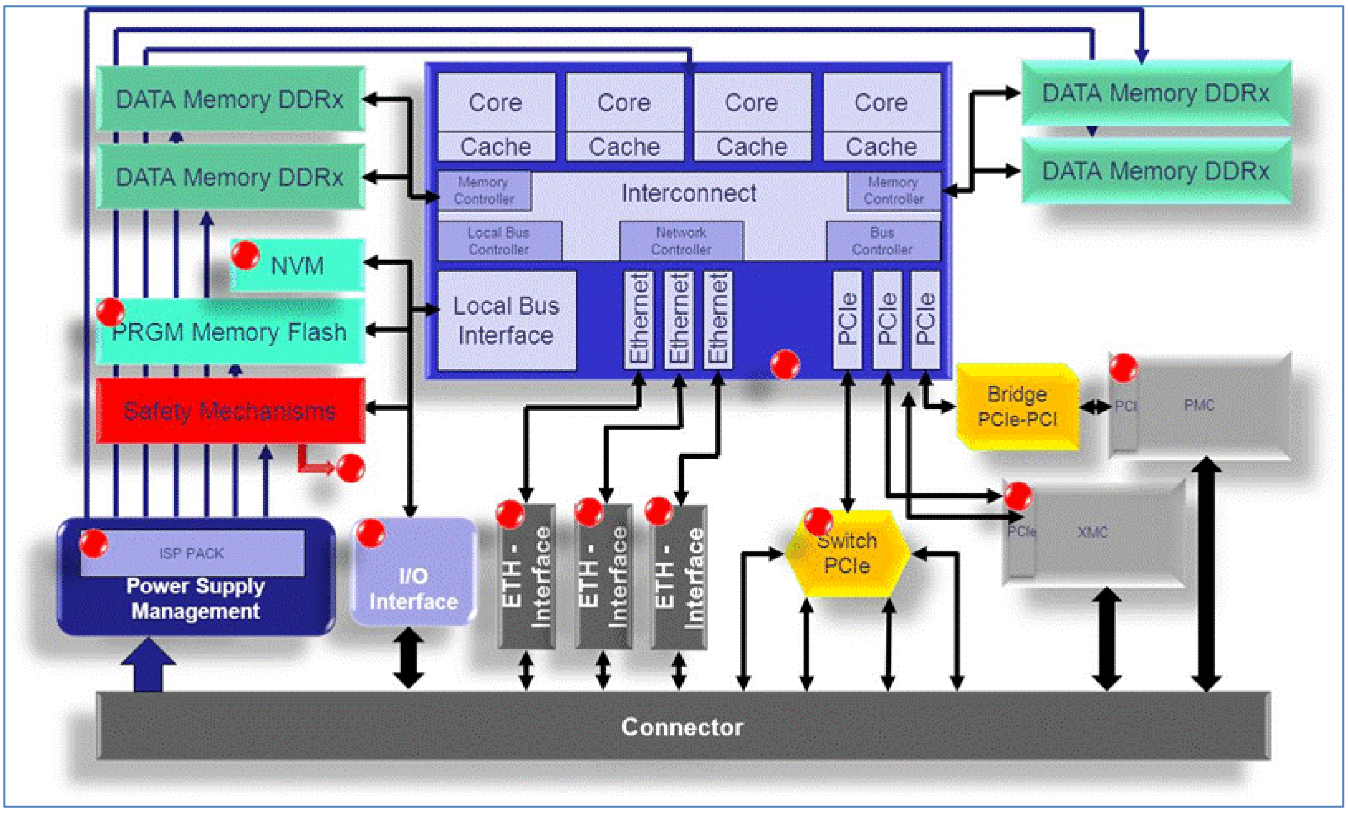


Figure 2: Typical Computer architecture.

Figure 2 shows in their environment the following elements that are described in section 5.3:

- A Microcontroller, which is in this particular case a multicore microcontroller.
  - Microcontroller are described in section 5.3.2,
  - Multicore microcontrollers in section 5.3.4.2.
- Different types of memories:
  - DDRx for data and program (see section 5.3.2);
  - Flash memories for program (see section 5.3.3.2);
  - NVM memories such as EEPROM, that are out of the scope of this report as they are considered as Simple COTS;
- A power supply with some SoC
  - Analysis has to be done to consider this programmable device as simple or complex depending on the numbers of Elementary Power Supply it can manage and control.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Some line interfaces (I/O, Ethernet, …) that are in general realized by a microcontroller or a PLD;
- A switch PCIe (see section 5.3.1);
- A bridge (see section 5.3.1);
- PMC (PCI Mezzanine Card) or XMC (Switched Mezzanine Card). These cards are in general COTS cards and are out of scope of the report.

5.2 COTS SELECTION

Considering the COTS devices involved in AEH generic embedded architecture, the present subchapter list the complex and highly complex COTS that are retained for the analysis. For each of them the character complex or highly complex is stated based on EASA CM - SWCEH – 001 [5]. A classification of complex COTS can be found in [6]:

- Microprocessors,
- Microcontrollers,
- Controller / Transceiver / Bridges / Switches,
- Graphic Processor (out of the scope of the study),
- Programmable Logical Device (PLD) and ASIC are not considered as COTS that are fully covered by the ED-80/DO-254.

The remaining COTS under study are:

COTS	Considered Examples	Status	Rational
Bridges	PCI/PCIe bridge	Complex COTS	Bridges contain wired logic configured by registers not exhaustively described in the available datasheets.
Particular interface drivers	ARINC 429 drivers MIL STD 1553 drivers	Complex	Except for some trivial cases (A429 single line transmitter or receiver used in a “word by word” protocol) that can be said simple, those communication interfaces have to deal with asynchronism and DMA access to shared resource leading to some complexity (In the past those functions are handled by ASICs or PLD instead of COTS)
DDR3 memories	Generic model	Complex Hardware item	DDR3 memories cannot be considered as complex due to their constitution but due to the non observability of the cluster memory-memory controller.
NAND Flash Memories	Generic Model	Complex Hardware item	NAND flash memories need firmware in order to manage wear, and complexity status is inherited from this lack of determinism to

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

COTS	Considered Examples	Status	Rational
			manage.
Complex Micro Controllers[12]	Freescale MPC8610 & generic model	Highly Complex Micro-Controllers	Microcontrollers implement complex peripherals such as PCI, PCIe, Ethernet, …
Multicore Micro controllers	Freescale P4080 with extension to other Freescale families.	Highly Complex Micro-Controllers	Multicore microcontrollers are considered as highly complex even if they would not implement complex peripherals.

Table 1: COTS List.

5.3 STATE OF THE ART OF EMBEDDED COTS

5.3.1 Bridges and switches

The terminologies bridges and switches often cover similar concepts of a device connecting two or more buses or networks[13] areas.

Three characteristics can be raised in order to separate both concepts.

(a) Technology homogeneity: In some application it appears that bridge applies to devices interfacing heterogeneous technologies (e.g. PCI to PCIe) and switches to devices interfacing homogeneous technologies (e.g. Ethernet).

(b) Parallel versus serial: Distinction between these two terms can be done also on the parallel / serial nature of the interfaced buses or networks. In this case bridges are dedicated to parallel buses and switches to serial ones.

(c) A third operational way to differentiate bridges and switches is to consider that:
- A bridge functionally connects two buses or networks that define two different worlds from technology (e.g. a PCI bus and a PCIe bus) or addresses point of views.
- A switch connecting *n* buses or networks of the same nature but add to the bridging function a switching function between the different connected buses or networks.

These three ways have advantages depending upon the point of view.

The criteria presented in former paragraphs can be summarizes in a table. For instance Table 2 presents examples of bridges and switches, considering combination of Technology homogeneity on both sides with the number of buses or networks connected on one side and the parallel or serial character on the other side.

[12] Not all Microcontrollers are qualified as complex. We consider here only complex microcontrollers

[13] The wording bus is reserved for communication media that address directly memory zones. Network is reserved for communication media that address apparatus that therefore manage their own memory addresses.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

	homogeneous 1 to 1	homogeneous 1 to n	heterogeneous 1 to 1	heterogeneous 1 to n
parallel to parallel	bridge PCI-PCI	no example	bridge PCI-local bus	no example
parallel to serial	Not applicable	Not applicable	bridge PCI-PCIe	no example
serial to serial	no example useless	switch PCIe switch Ethernet	Ethernet controller with PCIe interface (cannot be considered as a bridge nor a switch)	no example

Table 2: Classification of Bridges and Switches.

Note that in the context of OSI model [18] Bridges and Switches operate up to Level 2 (Data Link Layer)

5.3.1.1 Bridges

It exists two main categories of bridges:

- Homogeneous bridges linking two buses or networks of same nature. These bridges –consider for instance a PCI-PCI bridge on a PC motherboard – connect two address zones one on the motherboard and a second on a connected card (see for instance the example on Figure 3. If bus address map is common to both sides of the bridge (no overlapping address ranges), the bridge is said transparent. This is in general the case in avionics application in which the complete network is settled before bridge configuration. Here, the bridge does not operate address translation. In the PC example presented before the two memory zones have possible overlapping address, in this case the bridge is said non-transparent and in order to connect this two overlapping address zones, it has to ensure learning of connected device addresses and address translation.

- Hybrid bridges that connect in the same manner two buses of different nature for instance a PCI-PCIe bridge. In this case, in addition to the function ensured by the homogeneous bridge that remain, the bridge has a function of signal conversion that can become itself complex as in the case of translation between parallel PCI bus and a serial PCIe bus.

  Such a bridge is studied in section 8.2.

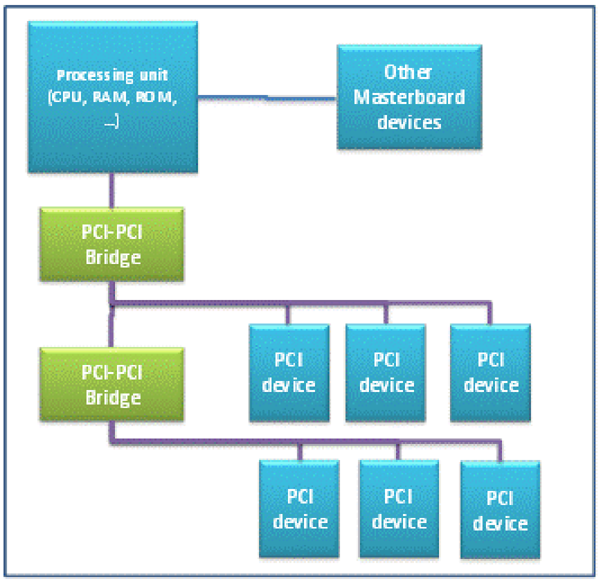


Figure 3: example of PCI sub-network architecture through PCI-PCI Bridges.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

5.3.1.2 Switches

In Table 2 it appears that switches are encountered in serial homogeneous 1 to n applications (e.g. Ethernet and PCIe). In these application switches allows interconnecting several buses or networks by dynamically switching of digital frame defined by a communication protocol.

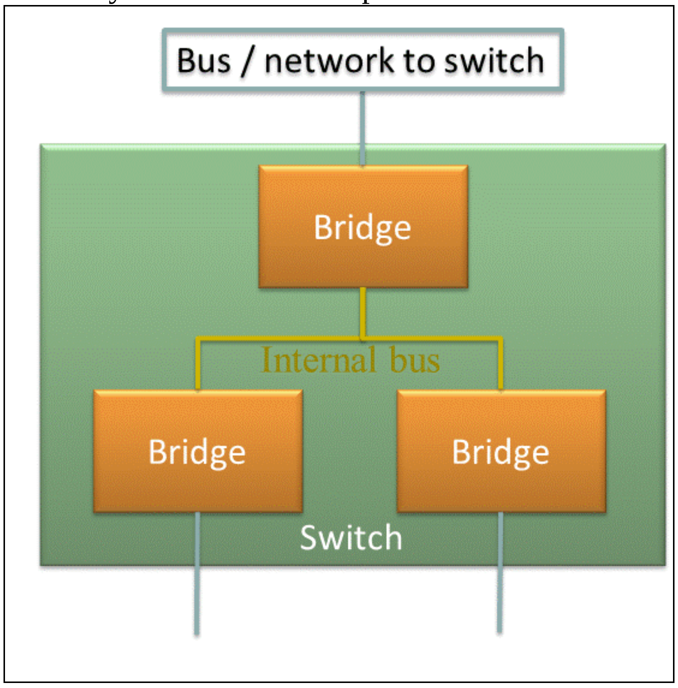


Figure 4: schematic view of a switch

Figure 4 presents a simplified view of a switch with bridges to connect the external buses or network to the internal communication bus that dispatch frames with respect to their addresses.
The major safety related application of switches in avionics is A664 (see Figure 5). In this domain the state of the art is non-COTS switches so that switches is not tackled here.

This Document Provided by the Abbott Aerospace
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

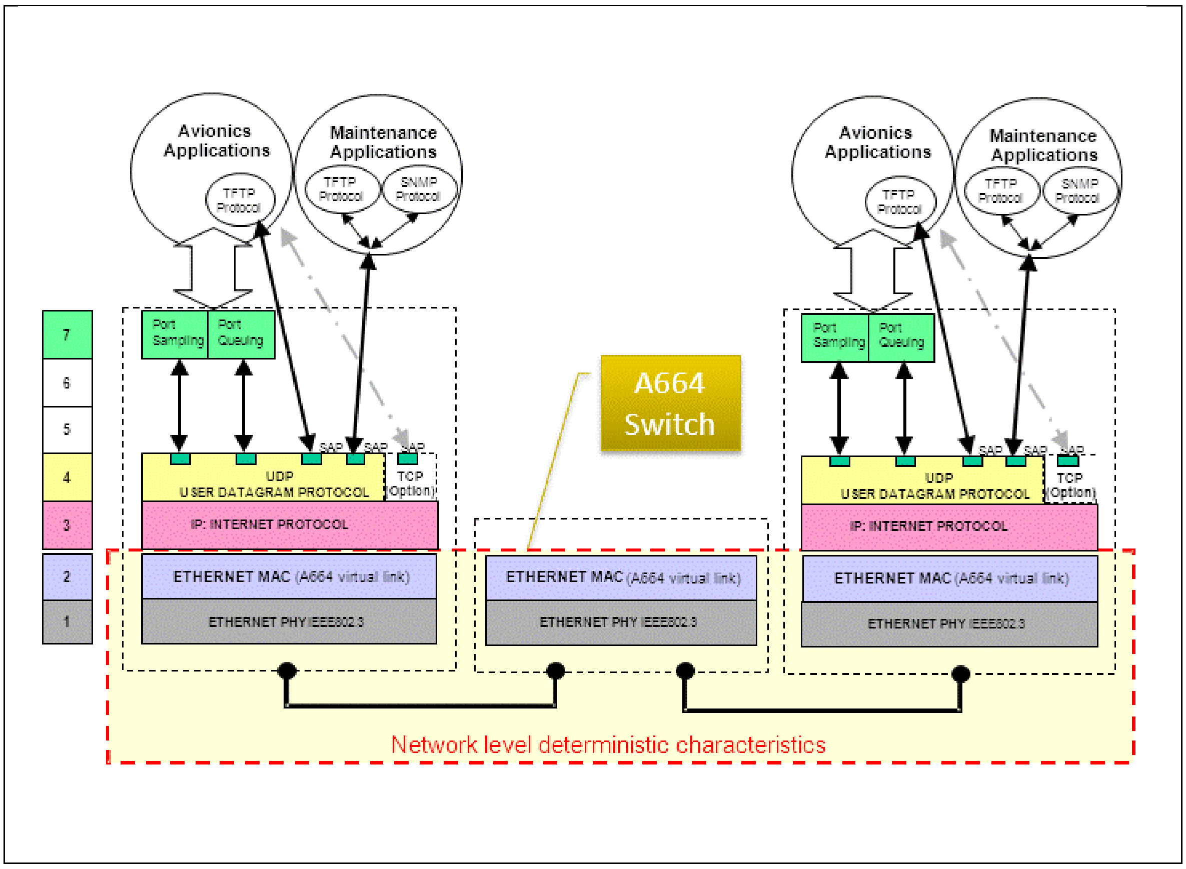


Figure 5: A switch in an A664 network.

5.3.2 Avionics Specific interface drivers

This type of component provides on one side connection to one low bandwidth normalized network type, on the other side connection to a processing core local bus either parallel or serial.

It can be seen as the low end type of the bridge family.

On the normalized network connection side, the component can provide down to the Physical Layer, including the analogue layers requested to adapt in reception and/or in transmission the external communication media.

On the processing core side, a large panel of local buses are available going from SPI bus solution up to PCI solution, while providing adaptability to microprocessor low end parallel local buses.

Amongst these avionics interface drivers avionics rely particularly on ARINC 429 and MIL-STD-1553 described hereafter.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

5.3.2.1 ARINC 429 interface drivers

ARINC 429 standard, firstly released in 1978, has been in service since early 80's for communication amongst avionics equipment and systems.

It is the most commonly used bus on commercial aircraft: Airbus A310, A320, A330, A340, Boeing B727, B737, B747, B757, B767, McDonnell Douglas MD11, etc.; and helicopters.

A429 have been for tens of years a typical domain for dedicated ASIC/PLD designs that were the most cost effective solution to implement such specific interfaces leading to a wide diversity of implementations.

Since few years, even if it is used uniquely in avionics context, COTS have been designed to integrate in a single component several services related to this communication standard.

A short description of ARINC 429 standard is proposed in subchapter 7.4. As the COTS studied are dedicated to the A429 decoding / encoding, the failure mode analyses are grouped in the COTS analysis chapter and in particular in subchapter 8.2.4.

5.3.2.2 MIL-STD-1553 Interface drivers

MIL-STD-1553 transmission standard was one of the first communication standard introduced in avionics in replacement of analogic communications. First released in 1973 the original standard, exclusively used in military applications (e.g. USAF F16), has been superseded in 1978 by version B (Notice 4 of version B has been released in 1996 [19]). It has been recently implemented on civil aircraft (A350) [20].

Beginning with COTS boards in the 80's, MIL-STD-1553 transceiver are since 2000's fully integrated COTS components.
Due to the MIL STD 1553B bandwidth (1Mbit/s) and the two redundant interfaced busses proposed on COTS components, either a parallel local bus or a PCI bus are commonly selected for processing core connection to cope with several megabit per second bandwidth. The latest generation of MIL STD 1553B COTS component also proposes core processing coupling through PCIe (see for example the DDC PCI-Express AceXtreme [21]).

The issue for the COTS manufacturer is more a small process size issue (see for instance Figure 6), with the need to provide a transformer insulation on the normalized network side and to mix in the same device analogue (differential, after transformer insulation) and digital (3.3V to 5V range) parts.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

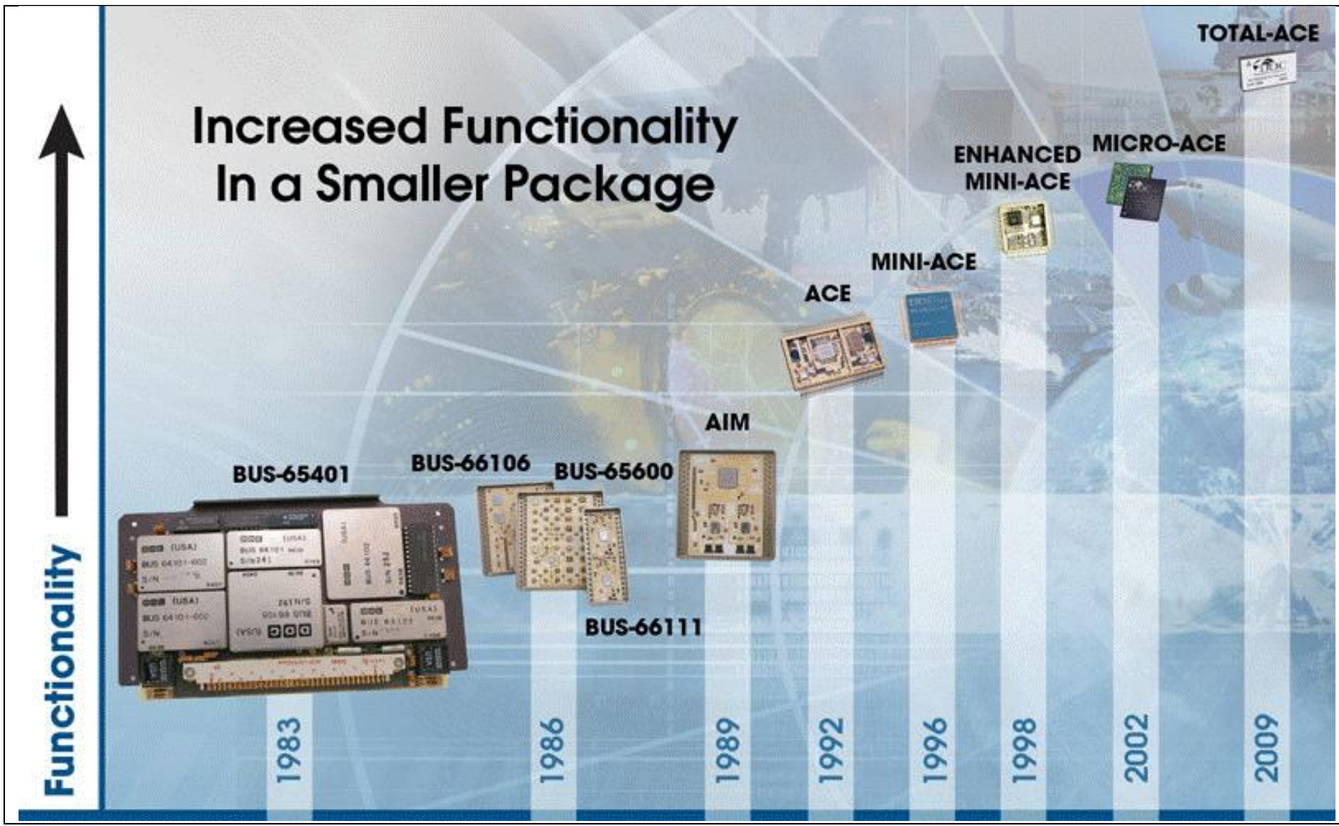


Figure 6: DDC MIL-STD-1553 driver evolution (from http://www.ddc-web.com/Images/New/Evolution.jpg).

5.3.3 Memories

5.3.3.1 DDR and QDR SDRAM

Memory chips have remained since many years the bottleneck in digital electronic performance. Moreover, if transition from SRAM (Static Random Access Memory) to DRAM (Dynamic Random-Access Memory) in 70’s showed a strong gain from economic and sizing point of views, allowing creation of large size memories and the development of computers, it has in the same time featured a loss in bandwidth performance and an increase in energy consumption[14].

Starting from these considerations, great progresses have been achieved in order to overcome the DRAM problems and in particular to increase their bandwidth[15] and to some extent reduce electrical power consumption. This led to a large variety of memory devices available today (e.g. DDRx, QDR, etc.). The Double Data Rate (DDR) Synchronous Dynamic Random-Access Memories (SDRAM) came along this way and are now the workhorses of the memory world.

Like classic SDRAM, DDR SDRAM is synchronous with the system clock. The big difference between these two memory technologies is that DDR reads data on both the rising and falling edges of the clock signal.

[14] Energy consumption and lack of bandwidth of DRAM is in large part due to the necessity to supply in energy in order to sustain the stored data and to refresh them periodically.
[15] Indeed for a long time the development axis of informatics has been the research of computational performances.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

When SDRAM only carries information on the rising edge of a signal, DDR module transfers data twice as fast.

DDR memories evolved from DDR2 to DDR3 and now DDR4. These different types distinguished by the depth of their prefetch buffers:

- DDR : 2 bits,
- DDR2 : 4 bits,
- DDR3 : 8 bits for higher bandwidth

I/O of these DDRx evolved too so that the voltage necessary for information transfer reduces from version to version allowing higher performances for lower power consumption.

In this context memory integrity relies on embedded memory controller. Memory transfer protocols are more and more complex and often request a dedicated memory controller (**Figure 7**). The observability of exchanges between the core and the memory is lost.

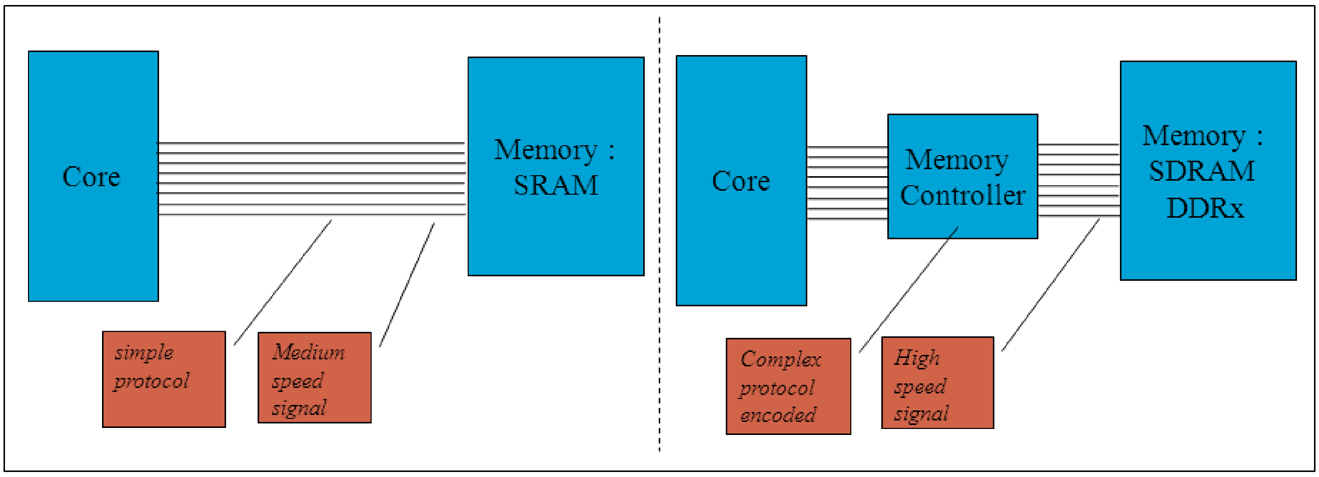


Figure 7: Memory access evolution between SRAM (no controller) and DDR SDRAM (complex controller).

<u>Note:</u> even in the cases where the memory stays outside of the microcontroller chip, the increase of memory frequency and the related signal integrity issues allow neither observability nor monitoring of the memory bus.

Due to differences amongst versions of DDRx, interface between microcontrollers and DDR memories evolves without full backward compatibility[16]. Latest microcontrollers as for instance Intel core i7 and Freescale QorIQ ™ devices are only interfaced with DDR3.

A single read or write access for the DDR SDRAM consists of:

[16] Difference in On die Termination on the board are another reason why different type of DDRx are incompatible each other. This point is out of the scope of the present report.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Single 2n-bit wide data, in one clock cycle is transferred at the internal DRAM core.
- Two corresponding n-bit wide data, in two-half clock cycle, are transferred at the I/O pins.

Figure 8 presents the detailed chronogram of a read action in a DDR technology. The clock (CK) is complemented by a clock (CK#) mirror of the first one. When command and address are transferred following CK, Data are transferred following CK and CK# so that with respect to CK there are transferred on both edges of the signal (double data rate).

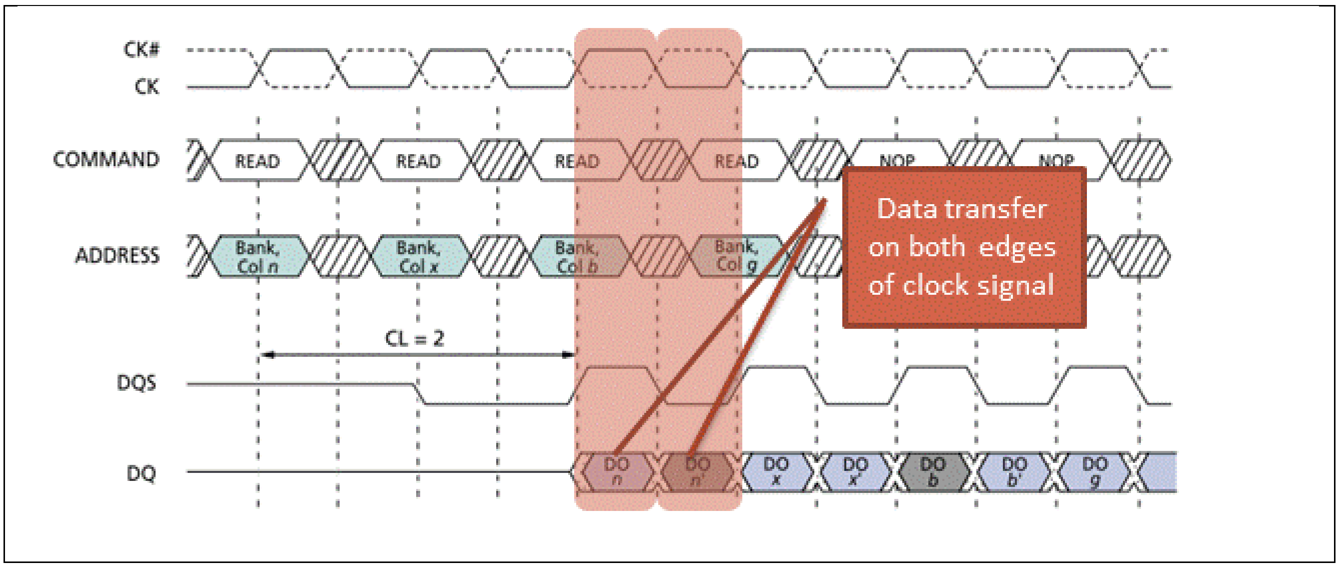


Figure 8: Transfer of data on both edges of clock signal in a DDR (source [22])

A simplified DDR / DDR2 architecture is given on following Figure 9. A more complete view details it on Figure 74. This architecture is not sufficient by itself to justify this chip to be complex. The main factor causing this classification is the quasi impossibility to observe the exchange between microcontroller and memory, due to high bandwidth and low applied voltage. Consequently, DDR memories can be considered from logical point of view as embedded in the microcontroller.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

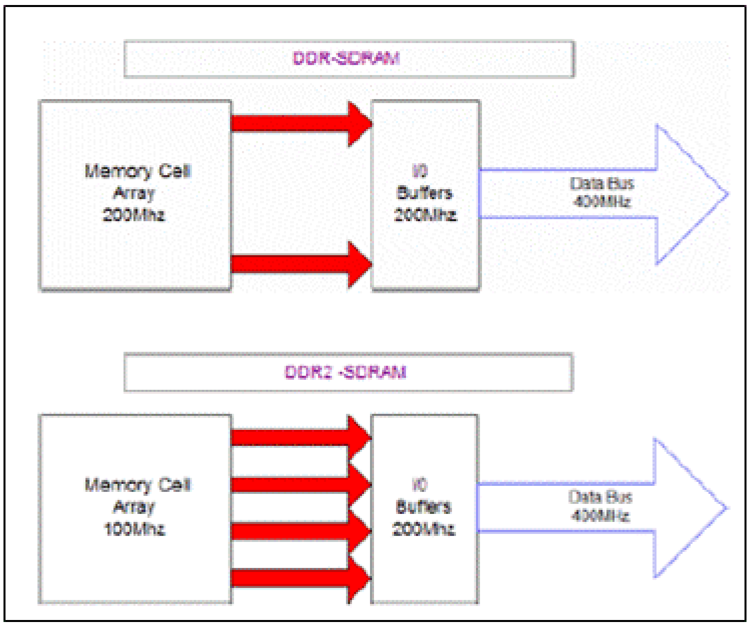


Figure 9: Simplified view of DDRx architecture.

QDR (Quad Data Rate™) SRAMs are a family of SRAMs developed in 1999 by the QDR consortium for High Performance Networking Applications. It is characterized by separate Inputs and Outputs that each operates at Double Data Rates.

Like Double Data-Rate (DDR) SDRAM, QDR SRAM transfers data on both rising and falling edges of the clock signal. QDR SRAM uses two clocks, one for read data and one for write data and has separate read and write data buses (also known as Separate I/O), whereas DDR SRAM uses a single clock and has a single common data bus used for both reads and writes (also known as Common I/O). QDR SRAM is not 2x faster than DDR SRAM but is 100% efficient when reads and writes are interleaved. In contrast, DDR SRAM is most efficient when only one request type is continually repeated, e.g. only read cycles.
Note: most SRAM manufacturers constructed QDR and DDR SRAM using the same physical silicon, differentiated by a post-manufacturing selection (e.g. blowing a fuse on chip).
As QDR and DDR SRAMs use same I/O technologies, the status about observability and monitoring is the same.

5.3.3.2 Flash Memories

Flash memories, in general, are an electronic non-volatile storage device appeared in the eve of 80's in Toshiba. It is based on the integration of transistors (floating gate transistors) [23] allowing a memory cell reduction per bit stored of 30%. It can be electrically erased and reprogrammed.

Flash memory now costs far less than byte-programmable EEPROM and has become the dominant memory type wherever a significant amount of non-volatile, solid state storage is needed. In the consumer market it is in particular associated with the development of CCD (Coupled Charge Device) Cameras, making obsolete the use (for these applications) of EEPROMs or battery-powered static RAM.

There are two main types of flash memory, which are named after the NAND and NOR logic gates. The internal characteristics of the individual flash memory cells exhibit characteristics similar to those of the corresponding gates (see Figure 10).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

NAND type flash memory may be written and read in blocks (or pages), which process size is generally much smaller than the entire device. The NOR type allows a single word to be written or read independently.

While non-flash EEPROM is erasable in small blocks, typically bytes, flash memory erasing is performed at large size blocks level (commonly named sectors). Flash memories do not offer arbitrary random-access erase operations.

In addition to being non-volatile, flash memory offers fast read access times. Although not as fast as static RAM. It is why the resident application software on high end processor is transferred at power up from the flash memory to the static RAM map for execution.

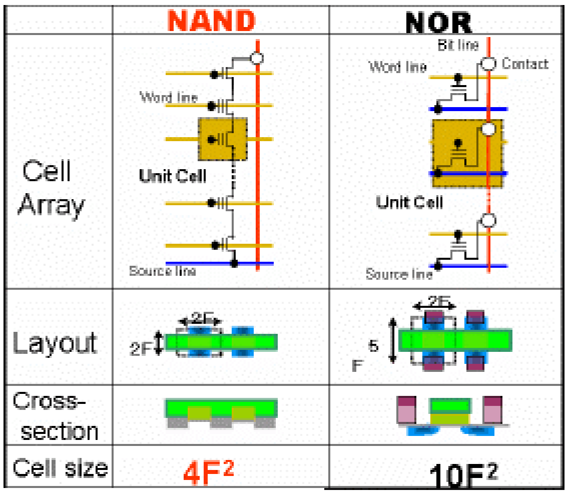


Figure 10: NAND and NOR Flash from [80]

The NAND type is primarily used in memory cards, USB flash drives, solid-state drives, and similar products, for general storage and transfer of data. NAND flash allow a denser layout and greater storage capacity per chip than NOR flash. NAND flash is typically permitted to contain a certain number of faults. Manufacturers try to maximize the amount of usable storage by shrinking the size of the transistor below the size where they can be made reliably, to the size where further reductions would increase the number of faults faster than it would increase the total storage available. NAND relies on ECC to compensate for bits that may spontaneously fail during normal device operation.

Common flash devices such as USB flash drives and memory cards provide only a block-level interface, or flash translation layer (FTL), which writes to a different cell each time to wear-level the device. Another limitation is that flash memory has a finite number of program-erase cycles. Most commercially available flash products are guaranteed to withstand around 100,000 Program/Erase cycles before the wear begins to deteriorate the integrity of the storage.

This effect is partially offset in some chip firmware or file system drivers by counting the writes and dynamically remapping blocks in order to spread write operations between sectors; this technique is called wear levelling.
Another approach is to perform write verification and remapping to spare sectors in case of write failure, a technique called Bad Block Management (BBM).

The method used to read NAND flash memory can cause nearby cells in the same memory block to change over time (become programmed). This is known as read disturb. The threshold number of reads is generally in the hundreds of thousands of reads between intervening erase operations. If reading continually from one cell, that cell will not fail but rather one of the surrounding cells on a subsequent read. To avoid the read disturb problem the flash controller will typically count the total number of reads to a block since the last erase. When the count exceeds a target limit, the affected block is copied over to a new block, erased, and then released to the block pool. The original block is as good as new after the erase. If the flash controller does not intervene in time, however, a read disturb error will occur with possible data loss if the errors are too numerous to be corrected by ECC.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

The NOR type, which allows true random access and therefore direct code execution, is used as a replacement for the older EPROM. NOR flash, as is used for a BIOS ROM, is expected to be fault-free.
For these reasons, some systems use a combination of NOR and NAND memories, where a smaller NOR memory is used as software ROM and a larger NAND memory is partitioned with a file system for use as a non-volatile data storage area.

Because of the particular characteristics of flash memories, it should be used with both, a controller to perform wear levelling and error correction, and specifically designed flash file systems, which spreads writes over the media and deal with the long erase times of NOR flash blocks. The basic concept behind flash file systems is the following: when the flash store is to be updated, the file system writes a new copy of the changed data to a fresh block, remap the file pointers, and then erase the old block later when it has time.

5.3.4 Microcontrollers

5.3.4.1 Single core

A Microcontroller is a highly integrated component containing on the same die: a single processing core, internal memory and programmable I/O peripherals interfaces and controllers.

Evolution of micro controllers has been outlined in section 1. Some examples of this evolution are

- The Freescale MPC7448, Intel Pentium II and IBM S/390 which are latest standalone processing core examples.

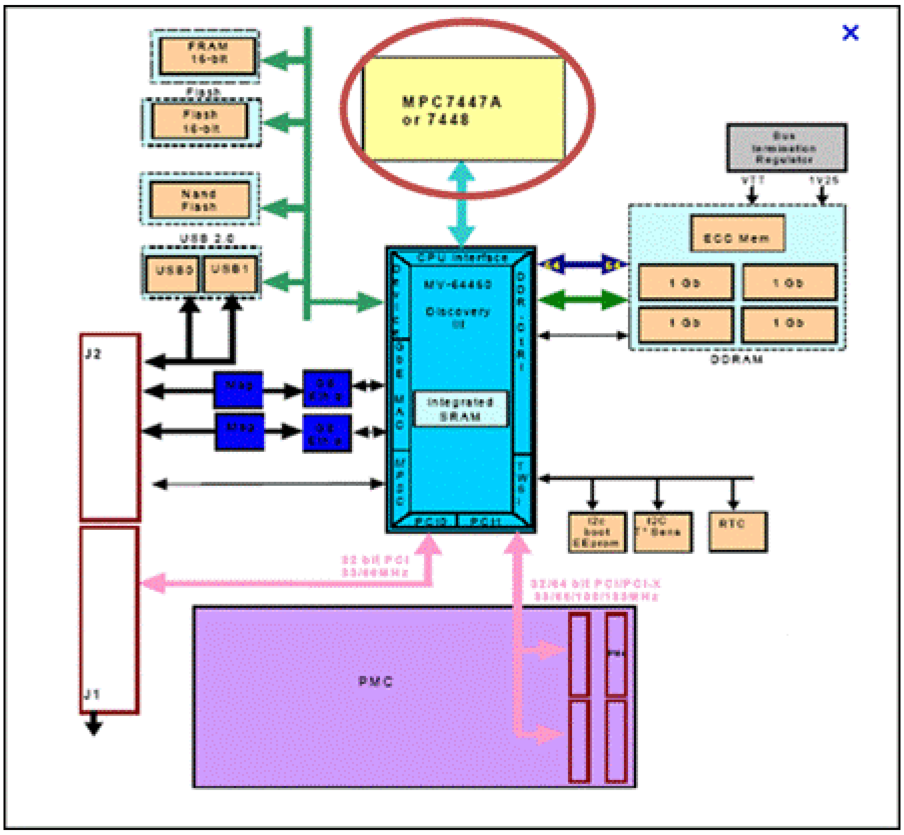


Figure 11: a typical board architecture around a MPC 4748.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

- Note: Reference [6] considered microprocessors as standalone complex COTS. EASA CM - SWCEH – 001 [5] recommended that "The development assurance of microprocessors and of the core processing part of the microcontrollers and of highly complex COTS microcontrollers (Core Processing Unit) will be based on the application of ED-12B/DO-178B [2] to the software they host, including testing of the software on the target Microprocessor / Microcontroller / Highly Complex COTS microcontroller.
- Simple Microcontrollers, that implement around a core simple peripherals, like UART, I²C, SPI. Examples are Freescale MPC5567, Texas Instrument C2000 Microcontroller series or the older NXP LPC2119;
- Integrated microcontroller with complex peripheral implementation and integrated memory controller like Texas Instrument DSP or Freescale MPC8610 (Figure 12).

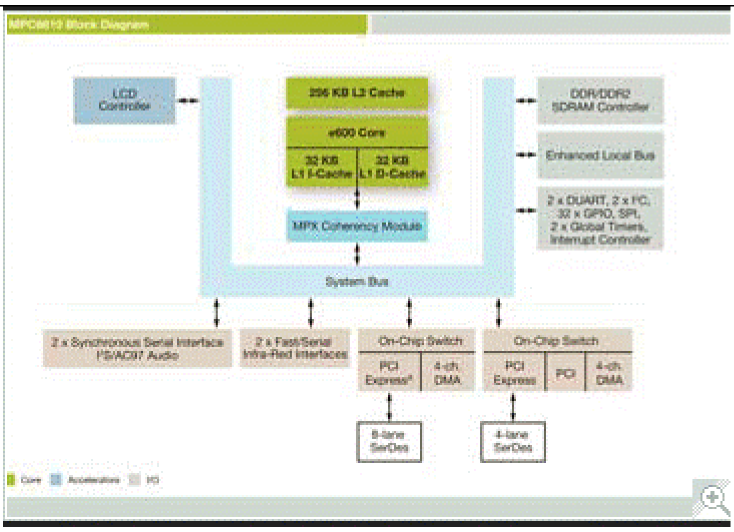


Figure 12 : Internal Block diagram for the MPC8610.

The integration of memories unit (like Cache) to large microprocessor cores implies that large amount of computation by the microprocessor are hidden to the surrounding system.

The representation of a microcontroller given on **Figure 1** does not represent any longer the typical architecture of a microcontroller.

A simplified representation of mono-core MCU architecture is given on Figure 13 (for example MPC 8610 or ARM AMBA family).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

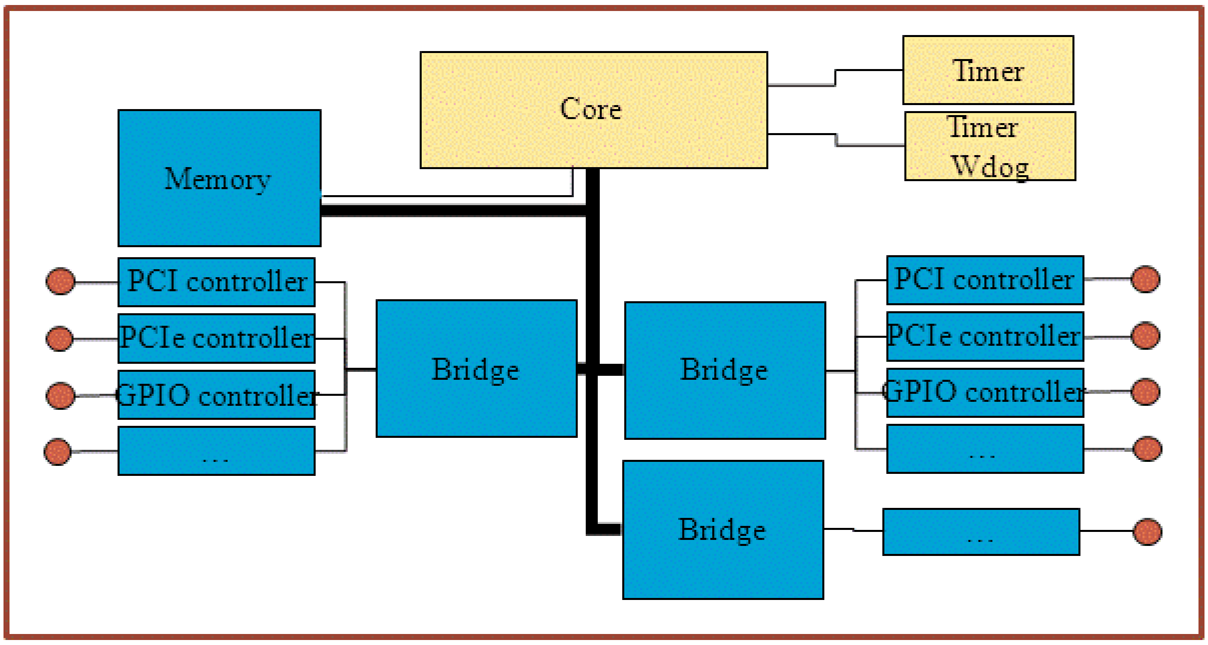


Figure 13: typical MCU architecture with internal bus

On most recent architecture, in general inherited from multicore MCU architectures, an interconnection module which generic name is “interconnect” distributes information to the different bridges - Figure 14.

On Freescale microcontrollers, for instance on P3, P4, P5 families and future T series, this interconnect is named CoreNet^{TM} [17], on ARM devices it is called CoreLink^{TM}.

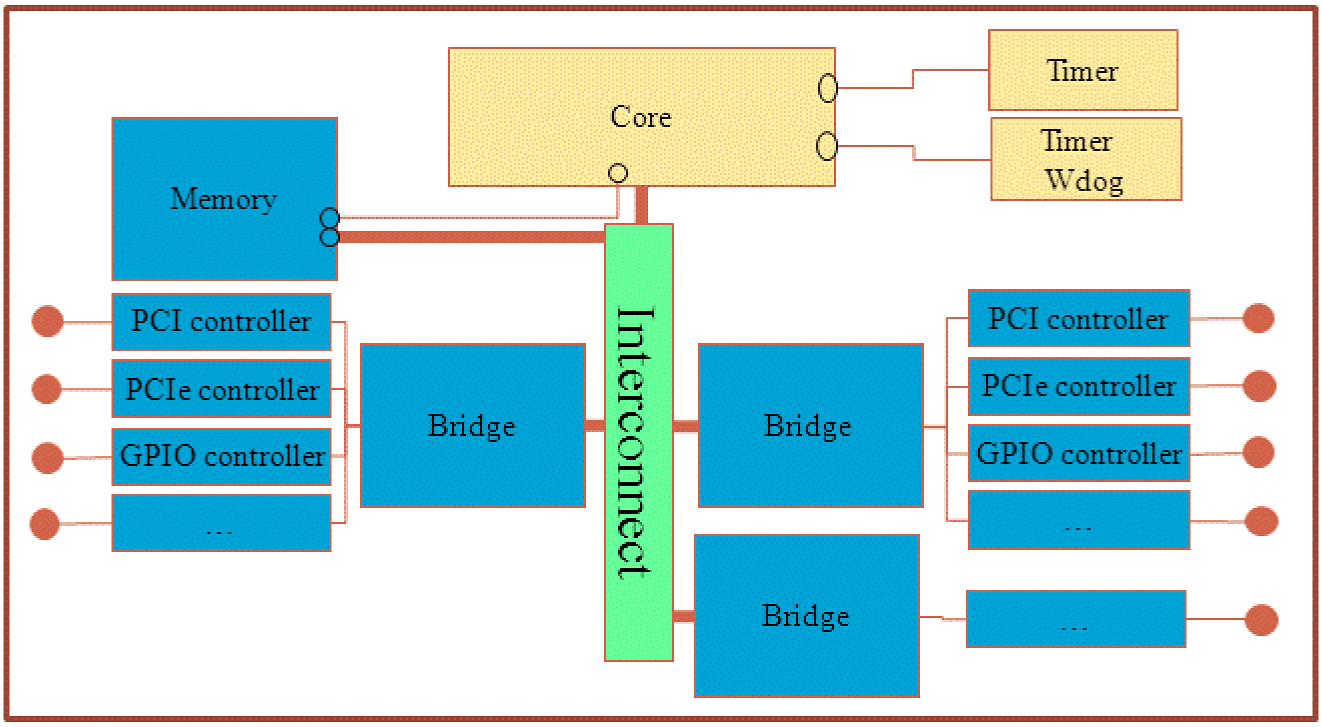


Figure 14: MCU architecture with Switch.

[17] CoreNet^{TM},OCeaN^{TM} and PAMU^{TM} are trademark of Freescale semiconductor. CoreLink^{TM} is a trademark of ARM.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Although, this interconnect structure is largely kept confidential by MCU designers, reference manuals analysis and tests show that it can be seen has a multi-switch module. Its structure is discussed in sub-chapter 8.8.

5.3.4.2 Multicore Microcontrollers

Since the introduction of the first microprocessor, the Intel 4004, in 1971 and for decades, progresses in microprocessor performance have been driven by lowering process size of chips, increasing the number of implementable transistors per chip (See Figure 15) and subsequently increase clock frequency[18].

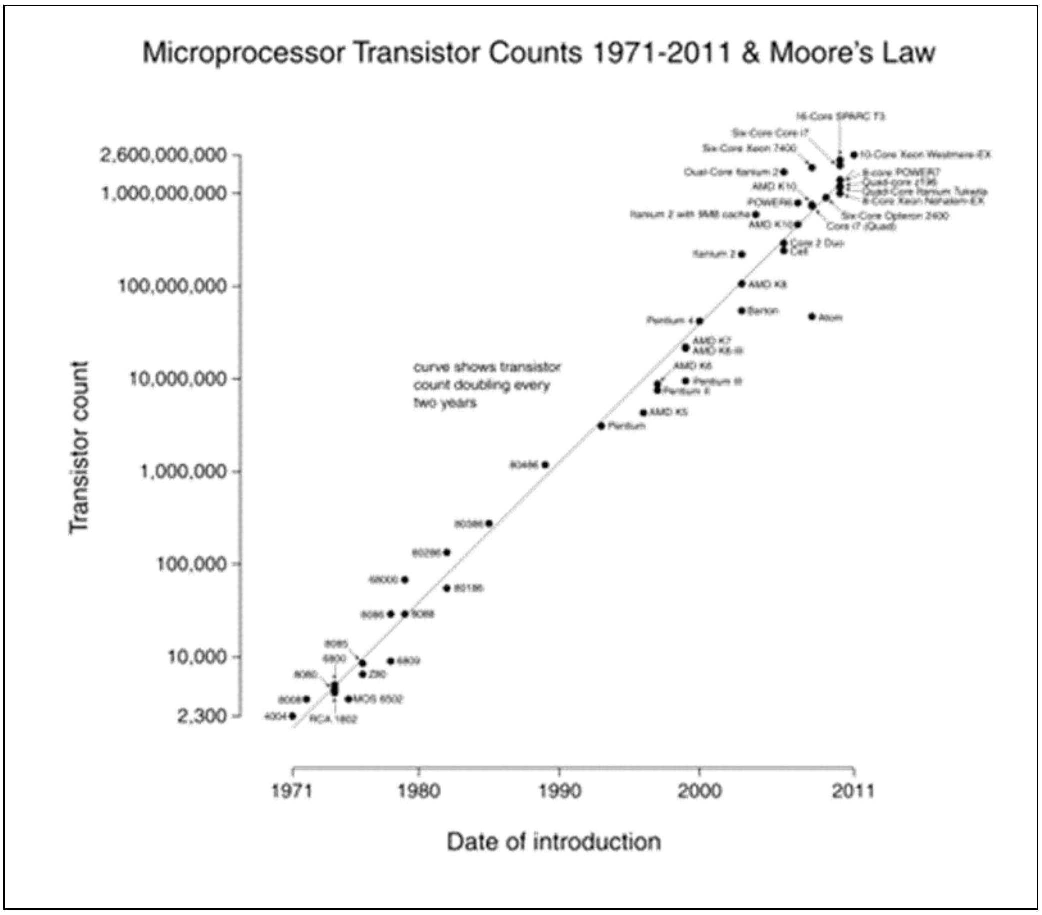


Figure 15: Moore law (src. Wikipedia [8])

It was anticipated that Moore laws will slow-down around 2013-2018 due to quantum effects:

- sub-threshold leakage which flows between the source and the drain of the transistor and which occurs before the transistor is on;
- gate leakage which flows through the thin gate oxide;
- Other leakages exist but are minor compared with both previous cases.

[18] The increasing amount of transistors enabled to design deeper pipelines and improved the embedded oscillators, which directly impacted the available clock frequencies on which the microprocessor can run steadily.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

For traditional manufacturing processes (2D planar technologies) such as the ones of TSMC, Global Foundries and other chip manufacturers, when the technology is scaling down in Deep Sub Micronics (DSM) process size domain, the quantum effects are increasing drastically. Nevertheless, new manufacturing processes such as the 22nm 3D FinFet technology of Intel and other 3D technologies are implemented to reduce or at least limit and better control these quantum effects for chip that reached 22nm in the last generation of Intel i7 in 2012

In fact before the process size shrinking reached the limit value, estimated around 4 nm, it appears that thermal dissipation is the first constraint directly linked to the clock frequency increasing of microprocessors[19].

Various ways were explored in order to overcome this limitation.

Before 2000s, the only criteria for the optimization were the performance and so all the technique / architecture enhancements were performed at core level.

- Increasing cache level 2 size in order to improve hit ratios and reduce the performance losses due to cache misses[20];
- Pipelining;
- Increasing the use of Instruction Level Parallelism (ILP) – pipelining in order to deliver one result per cycle after a latency time;
- Develop a full parallelism in instruction computing – Simultaneous Multi-Threading (SMT) – while applying ILP on each parallel thread. This technique has been used in processors from IBM (Power 5 and 6) and Intel;
- And others techniques as superscalar, branch prediction, speculative execution.

But due to the fact that firstly no more optimization was possible or found at core level and secondly that the power wall is reached (see note [19]) new solutions found at platform/SoC level have been embedding several cores on a single die. This solution is a natural extension of SMT with less design complexity.
The first multicore on the market was the IBM Power 4 in 2001. The present variety of multicore processors necessitates some classification.

[19] More precisely, the temperature increase of the chip is the visible effect of the increase of the power consumption. The power consumption must be divided into two contributors: the static power and the dynamic power. The static power is the leakage power. This power is increasing drastically in last SoC architecture (Power wall) because the technology scaling down. The dynamic power is due to the activity of the chip (proportional to Voltage^2 and to frequency) and is completely dependent upon the usage of the SoC resources.
[20] Increasing cache level 1 that is integrated into the deeper levels of the microprocessor is not possible without reduction of the clock frequency.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

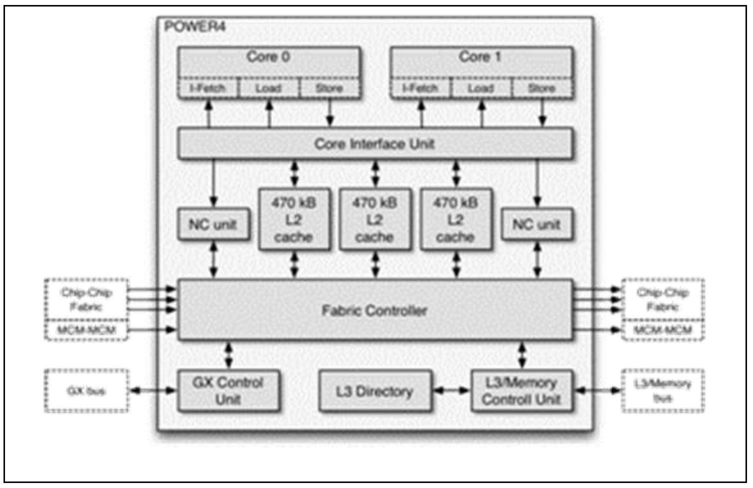


Figure 16: IBM power 4 architecture.

Moreover, since the 2000s, mobile platforms (essentially smartphones and tablets) are key drivers for the semiconductor industry and so a change of paradigm occurred where the optimization is not only seen from the performance point of view but also from the power consumption aspect.

The recent evolution in this domain encountered two steps:

(1) First these architectures focused the optimizations at SoC level by duplication of simpler identical cores on the same chip. The overall performance/power consumption ratio was consequently improved.

(2) Last, on mobile platform, chips evolve in order to become again heterogeneous with various kind of cores (such as ARM, Power, x86, etc.) and use of hardware accelerators in order to leverage the specificity of each one. Of course new problems occur such as for instance the software model to apply to those new architectures.

A first classification of current multicores can be driven by the homogeneous or heterogeneous nature of their cores. The majority of multicores embed replicas of the same core like the Intel core Ix or the Freescale QorIQ ^{TM} series like the P4080 based on 8 cores e500MC.
Some others have different cores with (e.g. the Sun Niagara 1) or without the same instruction set. Another example of heterogeneous multicore processor is the Cell (see Figure 17) produced by Toshiba, Sony and IBM in particular for the Sony PlayStation PS3. It involves a Power PC core for task distribution as main core and 8 specific cores (or co-processor) dedicated to complex computations. The 9 cores are interconnected through an “interconnect module”.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

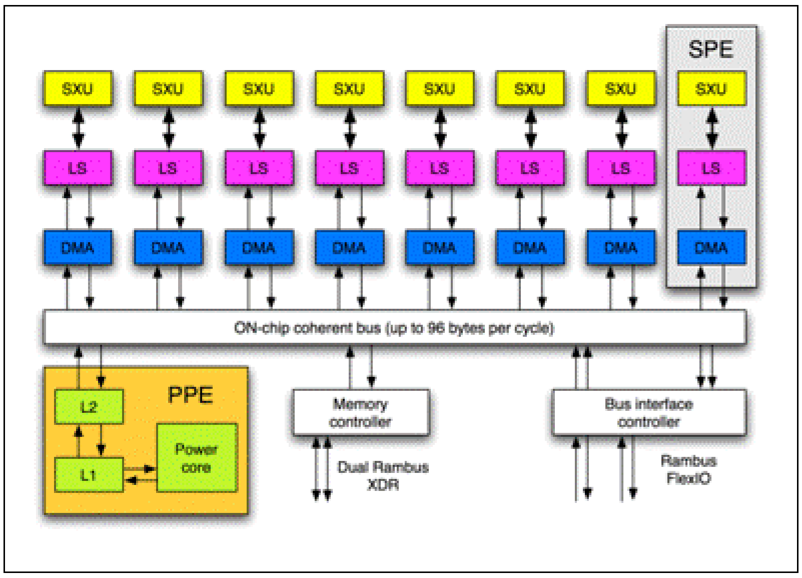


Figure 17: Cell: example of a heterogeneous multicore processor.

Homogeneous multicores are easier to design and to analyse, in another hand, heterogeneous multicores are dedicated to a particular application and more efficient in particular on energy consumption point of view.

This classification should not hide a particular feature of modern multicore processors which can have hidden cores realizing particular functionality independently from the other cores. A common situation that can be encountered is a multicore with N homogeneous cores that execute customer codes and few other cores that execute internal code in order to realize some function like accelerating a network access. Some multicores embed, in place of these extra cores, hardware accelerators[21]. These two types of feature are not always well documented and are even not always identified as complex IP. These complexes IP are sometime bought by the chip manufacturer to an IP provider.

A second axis of classification lies in the position of the L2 Cache. Indeed if the L1 cache is always attached to the core and separated within L1 instruction cache and L1 data cache, the L2 cache

- can be allocated to a single core (e.g. Freescale P50xx or P40xx families),
- allocated to a subgroup of cores (Intel core 2 quad),
- Shared amongst all cores (IBM Power 4 Figure 16).

With a L2 cache dedicated per core, the L1-L2, communication is local and interconnect is located below each L2 cache. With a shared L2 cache, interconnect is located between L1 and L2 cache. This can create non confidence in the L2 cache and lack of determinism. Local L2 caches are nevertheless more and more difficult to implement when the number of cores increases. This solution remains thus applicable up to 8 cores.

A third axis of classification lies in the nature of interconnection amongst cores and peripherals.

- Buses like Freescale MPX bus which can be compared at a first glance to a standard board bus with address and data burst sending is difficult to apply when the number of interconnected cores or

[21] We consider that a core is at least a processing element that implements an instruction set (ISA) available for the end user. A hardware accelerator implements an algorithm wired by hardware.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

peripheral increase. Such buses are no longer applied to multicore processors and are not applicable to more than few cores (link to arbitration impact on performances.

- Buses continue to be implemented in ring topologies.

Most of the multicore processors are in fact multicores microcontrollers. The interconnection means have to ensure communication amongst the cores but also amongst cores and the peripherals. The number of interconnected actors increases more than the number of cores and others technologies are needed.

- Crossbars type interconnection are applied,
- The most common approach in today multicores is an interconnect block (named generically interconnect and for instance CoreNet^{TM} on Freescale microcontrollers, CoreLink^{TM} on ARM microcontrollers, etc.) consisting of multi-switches network allowing to connect cores to each other, core to peripherals and DMA to memory controllers.
  Depending upon the number of connected actors, these interconnect switches can take various topologies at 1D or 2D.
- They can be the basis for more complex topologies for many-cores architectures with group of cores linked together with buses or crossbar and these clusters of cores connected together through an interconnect.

Memory topology around the core is another important concern of modern multicore processors. Memories were in general placed around the cores however the horizontals distances increasing the classical bottleneck constituted by memories become more and more important. It is tempting to stack memory on top of processors in order to use vertical proximity. For instance the Teraflops Research Chip introduced in 2007 by Intel is an experimental 80-core design with stacked memory. Major challenges in this domain are process integration and heat evacuation[22].

Other questions that can be addressed facing a multicore processor are the implementation or not of multiple threads per core or the extension of instruction set (see [24]). For example Intel has introduced in 2010 a dual thread - dual core processor that consist, for each core to sustain the execution of two threads simultaneously, the global performance is like the one for a 4 cores processor. This is also the orientation followed by Freescale for the T-series where the first product is 12 dual-threated cores with correspond to the execution of 24 threads in parallel.

The number of application hosted on a multicore microcontroller has drastically increased during these last 10 years. Adaptation to different customers to sustain all their needs – problem to be solved already for current mono core microcontroller generation – becomes more and more difficult to tackle.
This configurability of modern multicore microcontrollers has to be taken into account in the state of the art of these COTS even if it is not driven by purely physical or technical purposes but by commercial ones.
The configuration of microcontroller is done through registers that allows the customer to activate and tune the features offered in order to cover at best his needs. The extension of these register increases (around 5000 registers for the configuration of the P4080) producing complex configuration process and associated test. For instance, some registers are only reserved to the manufacturer, so that the impact of modifications

[22] For instance, this is already the case in mobile platforms,: the SoC (containing microprocessor, hardware accelerators, interconnect and IO controllers) and the memory chip (DDR2/3) are already stacked onto each other, and the temperature is monitored by the SoC chip to be sure that the memory chip will cross its max temperature junction 85 or 105 °C depending on the quality needed. The next evolution will be the WideIO memories where the silicon die of the memory will be directly connected to the silicon die of the SoC in order to get very short latencies and very wide bandwidth.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

of the values stored in these zones is unpredictable and non-described on the basis of reference manuals. Some other registers are hidden and can change drastically the behaviour of the COTS (that is in particular the case for interconnect configuration). This major concern is a key point in the control of modern multicore microcontroller usage.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

6 METHODOLOGY

6.1 BREAKDOWN LEVEL

Three main different breakdown levels can be envisaged.

1. A level in which the COTS under study is considered as a complete white box**.** At this level, the considered failure modes are those of internal detailed blocks of the COTS.
   - When describing the COTS at this level, it is possible to access a full description of the internal faults and theirs impacts on the output of the COTS.
   - Hence it is possible to define very precisely the faults that can be detected internally and the failures that have to be mitigated by architectural means.

   NOTE: Knowing that a full white box down to the gate level is never achievable, one difficulty is to determine which level of detail would be sufficient to allow adequate mastering of both the functional and dysfunctional behaviour of the COTS and reach acceptable coverage of potential internal failures or errors. See further considerations on that issue below in para.6.2.

2. A grey level in which the COTS is modelled with interconnected generic blocks**.** The COTS model is not fully representative of the component itself. This breakdown level is chosen sufficiently detailed in order to be heuristic and not too detailed in order to be free of property rights. In the present report this condition must be strictly respected because no document under NDA can be used. In an operational study this detail level can be lowered.
   Using generic blocks in the model allows identifying the blocks most relevant and generic faults that will cause failures, and the common causes to different failure modes.

3. An architectural level that considers the COTS under study as a black box. At this level, the failure modes are those of the output flows of the COTS. This breakdown level allows identifying a set of COTS failures and thus defining corresponding detection/ mitigation means. Indeed only failures perceivable outside of the COTS can be identified/detected/mitigated by architectural means.

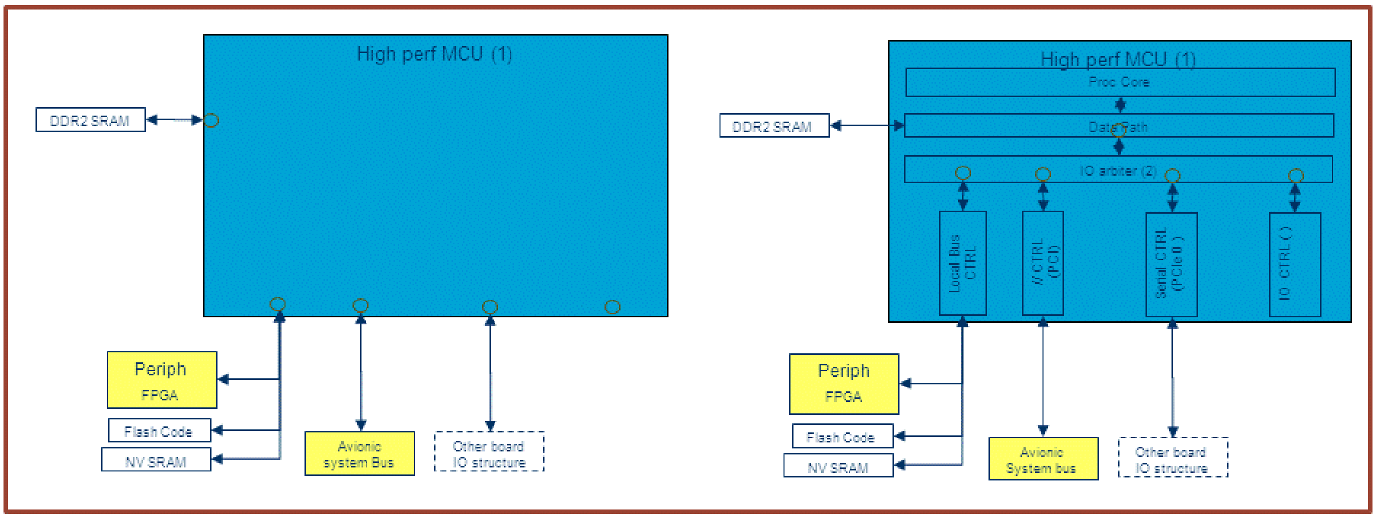


Figure 18: Black box (left) and grey box (right) point of view on a COTS. Note: the white box point of view has not been represented here.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

6.2 ABSTRACTION LEVEL

Independently of breakdown levels, a choice must be made on the level of detail (or abstraction) for which we consider the information exchanged by the COTS internally and with its environment.
Let us consider, for instance, COTS output signal transmitted on an external bus. There are basically three admissible levels on which this signal can be described:

1. Functional level. This level can be viewed from two points of view :
    a. System transfer function[23]: This point of view is not accessible in the present study that considers the COTS independently of any particular use. It has to be noted that system functional detection/ mitigation means are not directly addressed in the present report.
    b. Hardware transfer function[24]: for instance delivering a coherent signal to the different Software levels. These "electronic" functional detection/ mitigation means are addressed here.

2. Logical level. This level considers the output of the COTS from their global logical content point of view (For instance a 32 bits output signal). It is the most adapted level on which it is possible to catch the failure modes of the COTS and to identify the possible detection/ mitigation means.

3. Physical level. At this level, the COTS output are separated and signals are detailed up to physical characteristics (voltage, amperage, timing, etc.). This abstraction level is certainly too low for an effective description of failure modes on a complex COTS due to the number of I/O pins. Nevertheless it can be considered for some particular cases, in order to characterize a failure mode identified at higher level (for instance bit stuck at a value, voltage oscillations, timing shift, etc.) or in order to estimate the efficiency of detection/ mitigation means (for instance the monitoring of a signal shape). This abstraction level should allow also identification of failures which are due to the technology used (e.g. sensitivity to SEU - **S**ingle **E**vent **U**pset-, sensitivity to current or voltage fluctuation, etc.).

6.3 FAILURE MODES AT LOGICAL LEVEL

6.3.1 List of failure modes

At logical level, these failure modes can be classified into classes described below. This level deals with transfer of some information, encapsulated or not in a coherent message. Here "information" has to be taken in a general meaning for instance address, payload or control – see definition in section 3.2.2-. This transfer can be characterized by

- Three states that symbolized the transmission of the message that can be normally received loss or repeatedly untimely received (the corresponding faulty transition is described in the following paragraphs);

[23] In this report, "system transfer function" refers to the description of the applicative service provided by a hardware block, but more generally by complete COTS, to a functionally connected hardware block or to an environmental media.
[24] In this report "hardware transfer function" refers to the description of the service provided by a hardware block to connected hardware blocks or to an environmental media, for instance the software.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- A state diagram that refine at logical level the states and transition of the information carried by the received message. The states, symbolized on Figure 19 by $s_1, \ldots, s_n$, are in one hand admissible states - the logical admissible states of the information as it has been specified- and in the other hand forbidden states that should not be reached in normal conditions. The transitions of this diagram are, consequently, in one hand, the transition specified and in the other hand, some transition forbidden between two admissible states or between an admissible state and a forbidden state (see section 7.2.2 for an example).

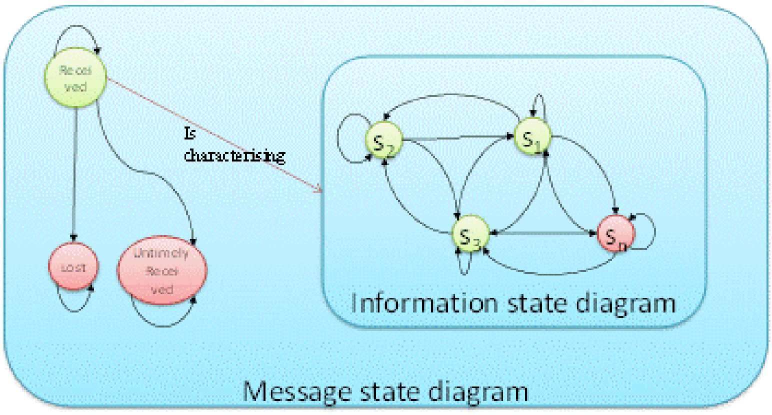


Figure 19: Example of a logical state diagram for information transfer. Although some transition have been hidden on the scheme, all are in principal possible unless some applicative restrictions.

Two possible events can plague this information transfer:

- A structural or temporal disturbance of the message encapsulating information. In this case it is impossible to enter into the information state diagram;
- A disturbance in the information state diagram, although the message is sent at the right time. In this case a transition between states is not realized, is realized untimely, or is realized between two states that should not be linked (forbidden transition). .

This classification is instantiated in the following categories (see [25][25]).

1. *Loss of message:*
   This mode corresponds to the absence of delivery of message when it should have been emitted as represented.

   This mode as an independent mode makes sense if and only if the energy state (high or low) that is reached when message is lost is not an admissible logical state of the information

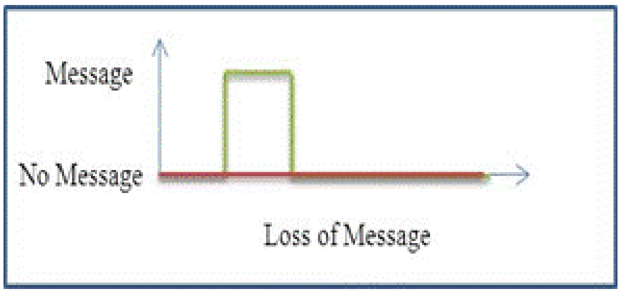


Figure 20: representation of message loss.

   If this condition is not realized, the loss of the message (for example following an open or short circuit on the transmission line) lead to a message that is interpretable by the receiver (for instance as a 0) we then prefer to assimilate this failure to an impossible transition of information to another state (described below).
   Note that this is a modelling choice that has no consequences on the final result.

[25] In [26] disturbances in transitions of the state diagram are supplemented by a third category – *erroneous transition from x to y when z is requested*. This type of failure mode "at solicitation" is not considered here for sake of simplicity and because it leads in general to the same detection means than *untimely transition* failure mode type. In other way, the untimely transfer of message was not considered because usually covered by other pre-existing design rules. Similarly abnormal sequence of messages was not considered in this reference because the technologies were robust to these failures.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Physical characterisation: If necessary the state reached will be defined as well as the potential duration of the information loss.

2. *Untimely transfer of messages:*
   This mode is two folds.

   Firstly, the untimely message transferred can be in advance. In the limit case of this advance message is systematics one can speak about babbling: receiver or transfer media saturation by extra communication. An example is proposed on Figure 21 where green line is the awaited activity and the squared signal is observed.

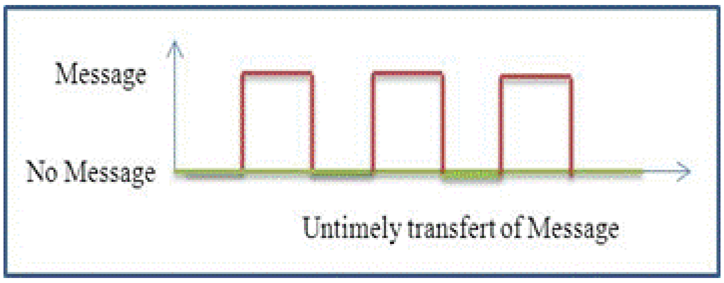


Figure 21: representation of untimely transfer of message.

   Secondly, the untimely message can be late. In this case one can speak about abnormal delay in information transfer. Figure 22 proposes an example, which dashed line the awaited signal and full line the observed signal delayed from the first one.

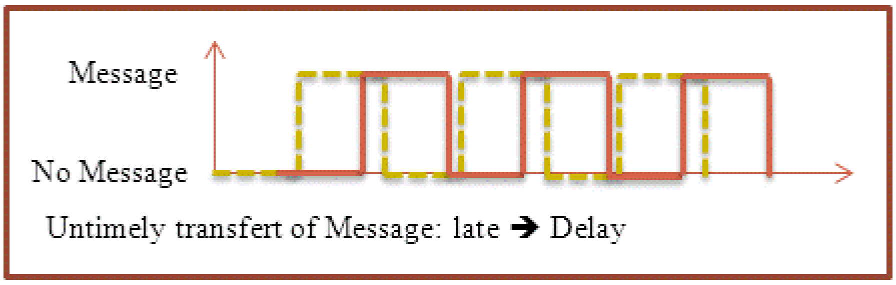


Figure 22: Untimely transfer of message (Delay)

3. *Abnormal sequence of messages*

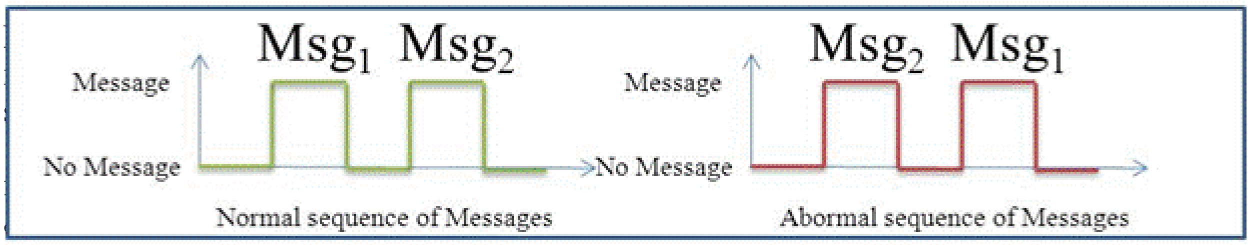


Figure 23: representation of abnormal sequence of messages.

   i
   l
   ure mode corresponds to an inversion in the receipt order of several messages (Figure 23). Consider for example two messages X and Y that should be transmitted and received in the order X, Y. An abnormal sequence of messages occurs if Y arrives before X.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

4. *Untimely or forbidden transition of information*
   This failure mode is also named information corruption or erroneous information transfer. If necessary, the duration of the error can be considered.

   In the particular case of information transferred with a protocol that considers the address of the transmitter, like ARINC 664 or ARINC 629, such a failure affecting this address can be particularized in "impersonation".

   When the transition is realised between two admissible states, the transition will be said "untimely". When this transition take place between <u>two states that should not have been connected</u> either because no transition was specified between these two states, or simply because the state of arrival is forbidden.

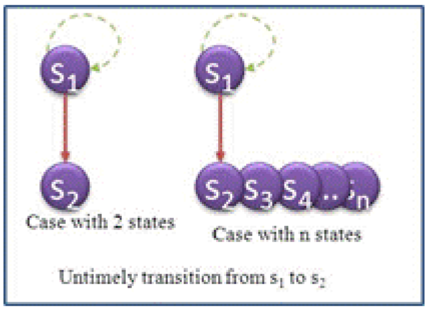

S_1
S_2
Case with 2 states
S_1
S_2 S_3 S_4 .. S_n
Case with n states
Untimely transition from s_1 to s_2

Figure 24: Representation of untimely transition between information states. The green dashed lines represent the requested transitions. The red empty lines the realized transitions.

5. *Impossible transition of an information*
   This failure mode can correspond to different categories: if it affects a part of an information burst it corresponds to an erroneous information transfer, if it affects a complete information burst or multiple information bursts, it correspond to an abnormal latency in information transfer. In this case the duration of the latency could be considered with respect to the typical fault tolerance time interval of the system;

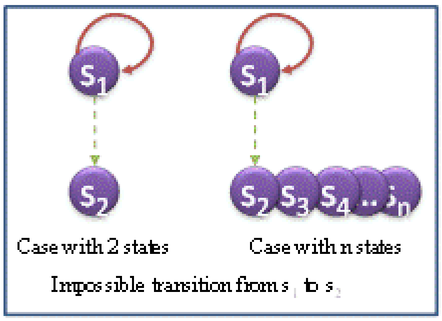

S_1
S_2
Case with 2 states
S_1
S_2 S_3 S_4 .. S_n
Case with n states
Impossible transition from s_1 to s_2

Figure 25: Representation of Impossible Transition between information states. . The green dashed lines represent the requested transitions. The red empty lines the realized transitions.

Relevant combination amongst previously defined failures can also be considered as they can reveal malfunctioning depending upon the technology of the external output or of the transfer to the software, these failure modes have to be adapted.

6.3.2 Comprehensiveness of the failure model

The failure model proposed is built to be comprehensive at logical level.
An entity whatever it is ensure some responsibilities with respect to its environment. These responsibilities are materialized on some outputs on which some messages are sent. At logical level all faults of the considered entity result into some failure mode of these output messages[26].
A message can encounter only the following modes:
- The failure of the message sending relatively to other messages;

[26] At physical level an internal fault may also result in Electromagnetic emission, heat, power over consumption, etc. that have only be considered marginally here.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Failure of the message sending itself;
- Failure of the message intrinsic constitution the message sending being correct.

The first category consists in abnormal sequences of messages, the second in untimely transfer of messages either delayed or advanced with a limit case in which the message is lost. The third category deals with messages sent in the right timing and order and which failure is on the information carried (data, addresses or controls). In this last category, information can change untimely to another valid value, to a forbidden value or to a valid value along a forbidden transition – this sub-category has been named untimely or forbidden transition – or information is frozen.
Following Figure 26 summarized this discussion and illustrates the completeness of the different cases.

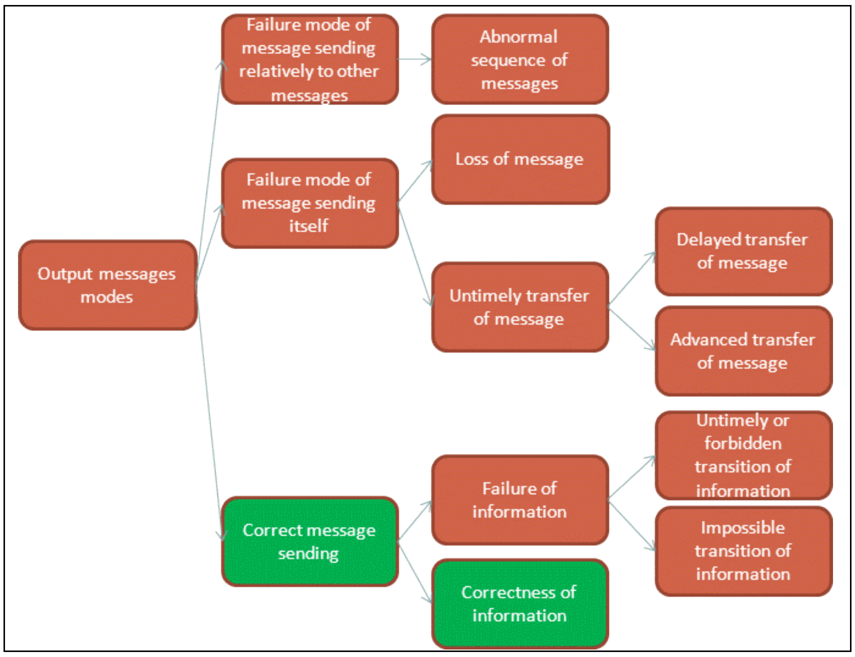

Output messages modes
Failure mode of message sending relatively to other messages
Abnormal sequence of messages
Failure mode of message sending itself
Loss of message
Untimely transfer of message
Delayed transfer of message
Advanced transfer of message
Correct message sending
Failure of information
Untimely or forbidden transition of information
Impossible transition of information
Correctness of information

Figure 26: Tree representation of failure cases

Note
The failure modes described in this subchapter can be linked to failure modes more commonly used. Table 3 provides an example of such correspondence.

Current failure model	Correspondence with commonly used failure modes
Loss of message	Loss
Untimely transfer of messages	Erroneously transmitted in time
Abnormal sequence of messages	Erroneously sequenced order
Untimely or forbidden transition of information	Erroneous data
Impossible transition of an information	

Table 3: correspondence between the current failure model and commonly used failures

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

The current failure model is chosen because of the ability to locate it at a given abstraction level and the ability to demonstrate at this abstraction level that the model is complete.

6.3.3 Particular failure modes at the hardware – software interface

When software is embedded by the COTS it has to be considered has an element of context for the COTS hardware (see Figure 27). COTS hardware has responsibilities with respect to this software as it has with respect to others external media. Considering the failure of these responsibilities leads to some new failure modes. Although these failure modes are inherited from the previously described failure model, it appears practical to particularize it.

The main responsibilities of hardware toward software are:

- Realize computation requested by the software;
  That corresponds to:
  - Get software instruction,
  - Get Data for computation,
  - Push computation results.

  The corresponding failure modes can be defined as:
  - No program instruction or data got,
  - Erroneous instruction or data got,
  - Latency in data delivery (Maximum Execution Time drift),
- When more than one application is embedded (in particular in an IMA framework) a new responsibilities appears that is to respect partitioning of the software application.
  In this case associated failures can be defined as:
  - Inversion of tasks.

  The non-respect of partitioning can also lead to already identified failure mode like
  - Loss or blocking of program instruction outing that is equivalent as "no program instruction outing",
  - Cross corruption of two software independent applications that is equivalent to erroneous calculation outing and
  - Latency in program instruction outing.

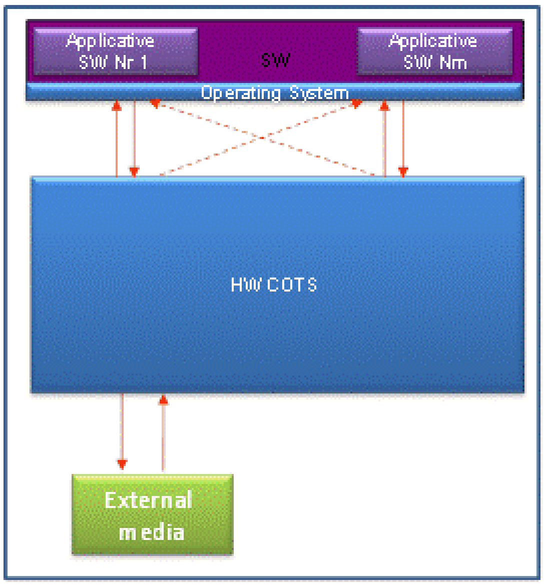


Figure 27 : General View on COTS with embedded SW. The lines represent flows that can failed, dashed lines represent flows that shall not exist. The partitionning symbolized here can be spatial or temporal.

6.4 FAULT CLASSIFICATION

Previous subchapter has classified the type of failure modes that can be considered for COTS outputs or COTS block interfaces. It appears also important to classify, errors at the origin of these faults and failures, from the point of view of their manifestation in COTS development process.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

ED79A-ARP4754A [1] mentioned the separation between systematic errors and random hardware failures and ED-80/DO-254 [4] separate the two concepts (see for instance the definition of fault in section 3.2.3) The separation of these two concepts has been theorized and extensively used in standard like IEC 61508 and associated standards:

3.6.5 Random hardware failure
Failure occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware. (IEC 61508 [26] part 4)

3.6.6 Systematic failure
failure related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors (IEC 61508 [26] part 4)

We associate SEU (**S**ingle **E**vent **U**pset) / MBU (**M**ultiple **B**it **U**pset) and other radiation initiated failures to random failures because they can be characterized by probabilities. Both Random Hardware Failures and SEU/MBU are out of scope of this study.

As suggested in the definition above systematic failures can be due to errors in the design, the manufacturing process or the usage (operational procedures).

<u>COTS design errors</u>

Even if state of the art is respected, some errors can be inserted in the COTS design, because of the complexity of the design process. These errors are typically:

- Non instantiate features,
- Untimely instantiate features (for instance non documented features that interfere with documented features),
- Erroneous behaviour of some features in some particular conditions uncovered by standard test patterns,
- Etc.

These errors are documented in errata list for those that have been already discovered and in certain conditions under dedicated NDA signed between the COTS manufacturer and the platform developer – these errors are considered as known by the community. Other errors are discovered by users during their tests. Due to the increasing constraints on time to market, the proportion of these errors, found by the platform developer, could tend to increase.

In general design errors do not age during product lifetime. They can thus be tested, discovered and mitigated before product entering into service.

<u>COTS manufacturing errors</u>

These errors can be for instance

- Pollution of silicon by tungsten during chemical vapour deposition,
- Micro cracks in wafer,
- Etc.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

Some of them are undetectable during design tests and can evolve (this is the case for impedance that may vary with time in case of pollution if impurity migrates). Burn in can detect some of these errors but many other requires additional architectural measures in order to be detected and/or mitigated.

COTS usage errors

COTS face usage errors when some divergences appear between intended and effective use. For instance a component specified for a stabilized voltage and facing during its life a strongly varying voltage, etc.

Notes:

- Effect of normal rate of ionizing particle (SEU, MBU, etc.) enter in the random failure category;
- Although it is not statistically random (it does not obey an exponential law), ageing of Deep Sub Micron components enters in the category of "random failure".

While usage errors are covered by specification robustness and dating, COTS random failure and manufacturing process systematic failure are covered by detection or redundancy mechanisms dedicated to safety critical behaviours.

The COTS design errors need to be systematically covered by mitigation mean, in order to guarantee feature behaviour. We will see in chapter 9 that this means can rely in some extent on test and in the following chapters on architectural mitigation means.

6.5 GENERAL PROCEDURE

6.5.1 General overview

Figure 28 shows the general positioning of COTS faults their propagation to system by direct impact or to system through their impact on software.

Based on this point of view the approach defined here has two folds, firstly a top-down approach, from COTS interfaces to COTS internal structure, and secondly a bottom-up approach from COTS internal structure to system.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

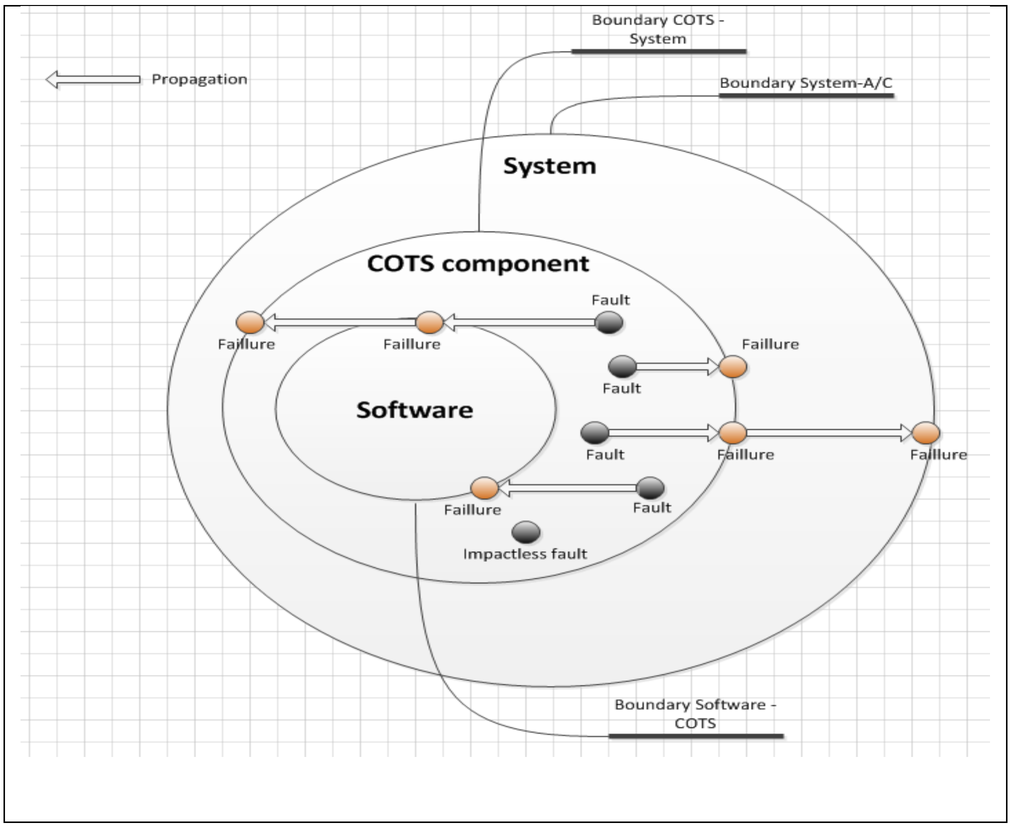


Figure 28: General positioning of COTS faults

6.5.2 Top - down approach

This first part of the analysis focus on COTS output failure modes. These modes have a particular position in the analysis as they are independent of the considered COTS internal architecture. The COTS may evolve to a new technology, if it has the same outputs, it will have the same output failure modes. Only failure occurrence may vary but this topic is out of the scope of the current study that focuses on systematic failures. Failure modes of outputs are also of great importance for establishment of Points of Observation, considering that internal exchanges of the COTS are not accessible.
This approach start in chapter 7 with output failure classification regarding the output technologies and the failure modes described in previous sections. For each interface technology, the existing detection and isolation means are listed. First causes in the COTS internal faults are then found. This top down approach is applied to the COTS selected in chapter 5.

6.5.3 Bottom - up approach

This approach aims at analysing internal architecture of COTS and determining failure modes of internal Blocks, their consequences on outputs and existing internal detection means. COTS internal architectures are deduced from analysis of COTS user manuals, reference manual and other COTS vendor documentations[27].

[27] Within this public study, no vendor documentation under NDA is used.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

COTS internal architecture is of great importance in order to determine the origin of failures observed on output flows. Depending upon the subject, the architecture considered in grey box models have been either the one propose by the reference manual or some simplified version of it (e.g. DDR, bridge).

Architectures proposed within reference manual can be a starting point for the analysis but when the usage domain is specified and some analyses are accumulated, it can appear that simplifications are possible for instance grouping of blocks or removal of unused blocks (see subchapter 6.1).

COTS internal block failures are defined on the basis of failure model already detailed in previous sections for COTS selected in chapter 5.

The COTS internal failure detection /mitigation existing mechanisms are accessible through the reference manual. The link is made with previous study on outputs, by determining fault impact on output failures. Faults that have no impact to the output failures are listed in order:

- For the study to be robust to changes of context;
- To identify the limits of the study.

This bottom-up approach is applied in chapter 8 to the COTS selected in chapter 5.

Both approaches top-down and bottom-up, are then complementary in order to cover most of COTS failures and define the best possible detection / mitigation strategy.

6.5.4 Failures Detection and mitigation

The first important means of mitigation of design failure is to detect them and mitigate them during tests so that they cannot occur in operation. This procedure is explored, in subchapter 9.2, in order to define zone in which block failure modes are tested with a sufficient level of confidence and blocks for which this level of confidence is not sufficient to rely only on test.

In parallel different detection mechanisms can be defined:

- Possible detection of output failures are:
    - Direct observations, by integration of Point of Observation, so that some failed output can be directly discriminated from normally functioning outputs. This direct observation request that some point of reference is given on the COTS output. These points of reference can be absolute (e.g. the comparison of an output value with a maximum admissible value) or relative (e.g. the comparison of an output value with a value calculated independently of the monitored COTS);
    - Controls and observations so that COTS is stimulated by a control signal emitted at some Point of Control and results checked at some Point of Observation.
- Possible COTS abnormal functioning can be monitored by COTS internal resources. This internal monitoring can be accessible and can be used to improved failure detections. These mechanisms can even be indirect measures of the global health of the COTS so that the abnormal functioning is not localised nor directly allocated to a given block fault.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Some of these mitigation suggestions are directly issued from the failure detection of previous step. Sometime, a more global view is needed: some detection / mitigation means do not reduced to direct observation or control/observation. It is for instance possible to control / observe several COTS with one mechanism or one detection means can rely on an external (system) mechanism to mitigate the error detected. These mitigation means is deduced in chapter 9 for the selection of COTS made in chapter 5.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

7 COTS INTERFACE FAILURE MODE ANALYSIS

7.1 INTRODUCTION

This chapter does not directly cover a task of the project as initially defined in chapter 0 but introduces preliminaries to COTS internal failure studies through a black box approach of the COTS with a study of its interfaces and associated failures. These failures are considered with respect to the logical level failure model, detailed in section 6.3. When necessary some causes at physical level have been considered in order to justify the possibility of a failure mode. Moreover for most complex interfaces (PCI, PCIe, etc.) some control information have been considered as enablers of other information (typically addresses and data), considering the failures of their supporting signals as causes for the failures of these considered information. It can be the case for instance of clock or reset signals but other control signals are also considered.

In the context of this study, as different families of COTS are studied, these interfaces are considered in full generality and without connexion to a particular COTS. Even though, on this black box approach can be branched a top-down approach considering

- That the output is associated to a COTS interface (for instance a PCI message is emitted by a PCI interface block) and
- That the failure of this message can be generated by this COTS interface block or by more buried blocks[28].

It pursues two purposes:

- Firstly, it presents the technologies and failure modes of interfaces that are the only ones on which an action is possible. By the way it exhibit also the failure mitigation mechanisms defined by the information transfer standards. It thus prepares to COTS internal faults (chapter 8) and Detection, localisation, mitigation mechanisms (Chapter 9);

- Secondly, failure modes and failure mitigation mechanisms of component interfaces are not only those of the transmission line but are also those of COTS blocks involved in this transmission. It is in particular important, when analysing a COTS Reference Manual to consider interfaces failure mitigation mechanisms as they are and not as COTS internal failure mitigation mechanisms. Indeed these mechanisms are designed to protect the zone between the mechanism encoding block, the transmission media and the decoding blocks of the receiver and not between the mechanism encoding block and COTS more buried blocks.

Although its interests listed above it is important to note that this approach does not allow detecting combined effect generated by the failure of a buried block contrary to a bottom up approach. Both approaches are thus complementary.

The outputs covered here are:
- Discrete interfaces,
- SPI bus,

[28] In next pages we consider that a “buried block” is an internal block of the COTS that is not directly interfaced with COTS context.

This Document Provided by the Abbott Aerospace
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

- PCI bus,
- PCI Express bus,
- ARINC 429,
- MIL-STD-1553

They have been organized from the simplest to the most complex without any tentative of classification between criteria related to

- Their protocol: buses or networks,
- Their connection: parallel – serial,
- Their extension: intra board, inter board, inter computing platform.

These classifications are recalled in Table 4.

	Protocol	Connection	Extension
Discrete interfaces	-	-	intra board, inter board, inter computing platform
SPI	Bus	Parallel	intra board
PCI	Bus	Parallel	Intra or inter board
PCI Express	Bus	Serial	intra or inter board
ARINC 429	Bus	Serial	inter board, inter computing platform
MIL-STD-1553	Bus	Serial	inter computing platform

Table 4 : classification of studied interfaces;

7.2 DISCRETE I/O

7.2.1 Description

Discrete interfaces of COTS are labelled in general in technical documents GPIO (General Purpose IO). They obey Transistor-Transistor-Logic (TTL) standard and have their logical low level between 0V and 0.5V and their logical high level between 2.4V and 5V.

GPIO interfaces are to be considered as simple interfaces. Their study corresponds to the first purpose defended in introduction of chapter 7.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

7.2.2 Failure modes

The failure modes of a GPIO are classified through the categories exposed in subchapter 6.3.

As already outlined in this subchapter, the "*Loss of message*" cannot be considered as acceptable mode as the lower energy state (0 volt) will be interpreted by the receiver as valid information.

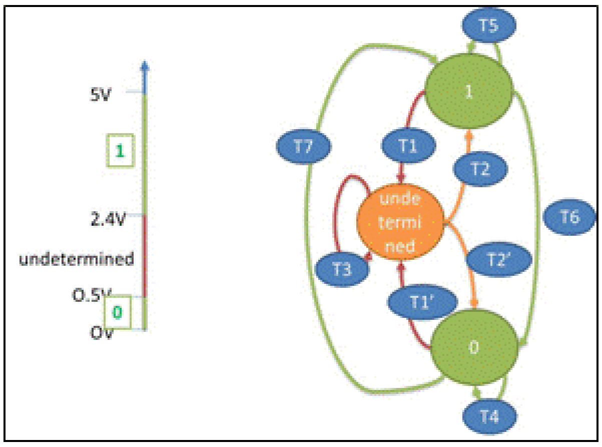


Figure 29: GPIO undetermined band and corresponding logical level state diagram;

Similarly, the mode "*untimely transfer of message*" cannot be considered as valid mode as a GPIO as always a physical value even when no information is transmitted through this value.

The failure mode *"abnormal sequence of message"* is included in this case in the mode *"Untimely or forbidden transition of information"* as sending for instance 10 in place or 01 can be seen either as an abnormal sequence or as an untimely transition from 01 to 10.

In the case of GPIO a general state model can be constructed (Figure 29) considering that basic information is one bit that can have a value 0, 1 and an undetermined state between 0.5V and 2.4V that can be erratically interpreted by the receiver as a 0 or a 1.

Figure 29 shows on the left side the undetermined voltage band and the logical states associated to the different value, and in the right side the corresponding state diagram at logical level.

On this diagram the transitions:

- T1, T1', T2 and T2' are *"Forbidden transition of information"* to or from a forbidden state than can lead to erratic interpretation of information by the receiver;
- T3 is an *"Impossible transition of information"* that stay in a forbidden state;
- T4 and T5 can be functional stable values or *"Impossible transition from a functional state"* and
- T6 and T7 can be normal functional transition or *untimely transition to another functional state*.

The transitions T1, T1', T2, T2 and T3 can only be generated by the output stage or also possibly by the power supply that can lower the upper level in case of loss of power. They are not discriminable from normal functioning state except in case of output physical monitoring. This monitoring is difficult to design and considering the previous cause analysis and the simplicity of GPIO output stage a monitoring of power supply can efficiently replace it.

The transitions T4, T5, T6 and T7 can be generated by many different COTS blocks. As already note they are not discriminable from normal functioning on the basis of logical state diagram alone. Ways to discriminate these failures from normal functioning can be:

- Distortion in the shape of the signal at physical level. This monitoring is difficult to design.
- Comparison between the logical signal and a reference signal. The comparison can be done
  - with an external information (spatial detection);

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- With a calibrated periodic signal (for instance a calibrated sequence of 0 and 1 at power on and/or power off) in such a way that receiver can make comparison between the receipt and what should have been sent.

7.2.3 Intrinsic failure mitigation mechanisms

No intrinsic failure mitigation mechanisms are defined on GPIO. Subsection 9.3.4.1.1 lists some potential mechanisms implementable.

7.3 SERIAL PERIPHERAL INTERFACE (SPI) BUS

7.3.1 Description

SPI is a simple interface that allows one chip to communicate with one or more other chips. It is

- Synchronous,
- Serial,
- Full-duplex,
- Not plug-and-play,
- There is one and only one master, and one (or more) slaves (see Figure 30) for more details:

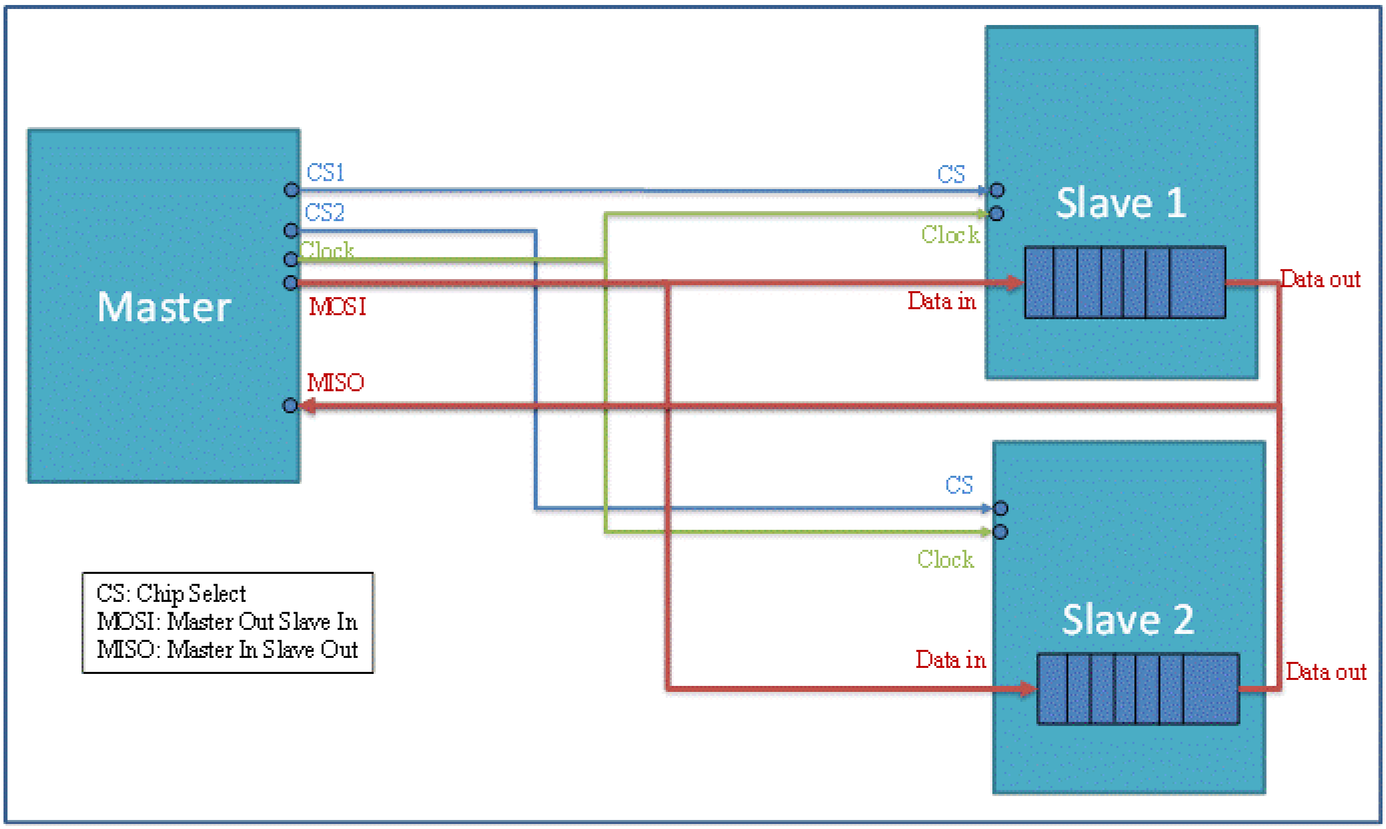


Figure 30: SPI bus basic principle

Figure 30 presents the way SPI function in a typical star type topology. It should be noted that a Daisy Chain type topology is also possible in some particular application (as JTAG) with the slave 1 MISO used as slave 2 MOSI.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

The transmission is cadenced by the clock signal generated by the master. On Figure 30, it can be noted that there are two wires for data (MOSI and MISO), one for each direction, so that

- Data is serialized before being transmitted,
- One bit of data is transferred on each data wire each time the clock toggles (full duplex and synchronous communication).

The communication is always initiated by the master that selects the receiver slave through CS (Chip Select) signal. The detail of the communication (bit order, length of data words exchanged, etc.) is particular to each slave and encoded in master registers.

At physical layer each interface (CLK, CS, MISO, and MOSI) can obey TTL or at least obey the same constraints: a voltage range corresponding to 0 logical value, a voltage range corresponding to 1 logical value and an undetermined range between both.

SPI interface is also considered as a simple interface. As GPIO, their study corresponds to the first purpose described in introduction of chapter 7. The analysis of SPI can be easily extended on the same basis to other simple interfaces like UART or I²C.

7.3.2 Failure modes

In the present chapter we suppose that the Complex COTS under study is the master. In case of complex COTS under study is slave of another complex COTS, many of the following failure are non-applicable.

7.3.2.1 Loss of message

Only CLK can be lost because the state of lower energy (0V) does not correspond to any admissible physical state. The cause of Clock loss can be the COTS Clock loss or a failure of SPI driver.

7.3.2.2 Untimely transfer of message

SPI transfers information at each clock transition so that, untimely transfers of CS or Data are equivalent to "*Untimely or forbidden transition of information*" described in a following subsection, except if we consider a transition of SPI module Clock. In this scenario, the master induces an information transfer between two clock pulses. This kind of failure can be due to the master clock, to the SPI clock through a corruption of the SPCON (Serial Peripheral Control Register) register. A priori no buried blocks can be involved except the clock. Then three cases can occur:

a. Untimely transfer of one chip select signal,
b. Untimely transfer of multiple chip select signal,
c. Untimely transfer of data signal.

In cases –a– and –b– correct data can be sent to the wrong register(s).
In case –c– data may be lost or sent to the wrong register as they are transferred before normal transition of chip select.

7.3.2.3 Untimely or forbidden transition of information

a. *Untimely transition of chip select*
One chip is selected in place of another. This kind of failure cannot be simply discriminated

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

from a normal functioning. It can be caused by the SPI driver itself, the transfer line or more buried block of the COTS.

b. *Forbidden transition of chip select:* this case corresponds to a chip select that is not in the list of chip. By the way, no chip is selected and the information will be lost.

c. *Untimely transition of multiple chip select*
Many chip signals are selected simultaneously in place from others. Like the first one this failure cannot be simply discriminated from a normal functioning. It can be caused by the SPI driver itself or by more buried blocks in the COTS. The probability that the transfer line cause such a failure is very weak and can be excluded except in case on board design error out of the scope of the present study.

d. *Untimely transition of data*
One bit or several bit changed untimely given an erroneous coherent new data. This failure can be caused by SPI driver or transmission line but more probably by more buried blocks in the COTS master or slave. Master can corrupt MOSI data and Slave MISO received data. This type of failure is not detectable at logical level because this transition is similar to normal transition generated through normal functioning.
Detection of this failure mode type needs a reference point. For instance time redundancy of sent data will basically detect failures of SPI driver or downstream block (in general SerDes – serialiser-deserialiser block); redundancy by a data generated by the COTS can detect failures of the internal datapath of the COTS, and downstream blocks including SPI driver.

Data cannot suffer a forbidden transition as, at this logical level of abstraction, all data values are possible.

7.3.2.4 Impossible transition of an information

a. Impossible transition of clock signal (clock frozen)
This failure can be due to COTS clock failure or failure of SPI driver or downstream elements like SerDes. It is detectable by a slave that can verify the vivacity of the communication, namely that this communication is still alive.
b. Impossible transition of one chip select
It is impossible to change the chip selected for communication.
c. Impossible transition of multiple chips select.
It is impossible to change the multiple chips selected for communication.
d. Impossible transition of data
Erroneous data sending.

Causes of these three kinds of failure and ways to prevent them are similar to those of "*untimely or forbidden transition*". If the impossibility of transition is permanent a vivacity check can be implemented. Its implementation at applicative level allows covering failures of more buried blocks.

Note: in this particular clock loss of SPI driver, chip select or data, correspond to impossible transition of these data.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

7.3.3 Intrinsic failure mitigation mechanisms

There are no real intrinsic failure mitigation mechanisms in SPI protocol.

However, some slave devices are designed to ignore any SPI communications in which the number of clock pulses is greater than specified. This measure is not general and some slaves only ignore extra inputs and continuing to shift the same output bit.

When implemented, this mechanism can partly protect against "Untimely transfer of data signal" or "untimely or forbidden transition of data" when it occurs between two clock pulses.

Some COTS manufacturers specify some parity bits in their implementation of SPI standard (see for instance Texas Instrument TMS320DM36x microcontroller user manual). These parity bit mechanisms are encoded in the SPI driver and cover only the SerDes and the transmission line. They of course request in order to be effective that the slave codes parity bit on MISO signals and decode parity on MOSI signals.

7.4 ARINC 429

7.4.1 Description

The ARINC 429 specification, defines electrical characteristics, word structures and protocol necessary to establish bus communication.
The bus structure is founded on a single transmitter – or source – connected to N receivers (N being 1 to 20) on a communication media consisting of a twisted wire pair. Because of this very rigid structure, data can be transmitted, on each communication channel, in a unique direction only (communication is qualified as simplex). Bi-directional transmission requests two channels or buses.

The most common ARINC 429 bus topologies are star or bus-drop ones (see Figure 31).

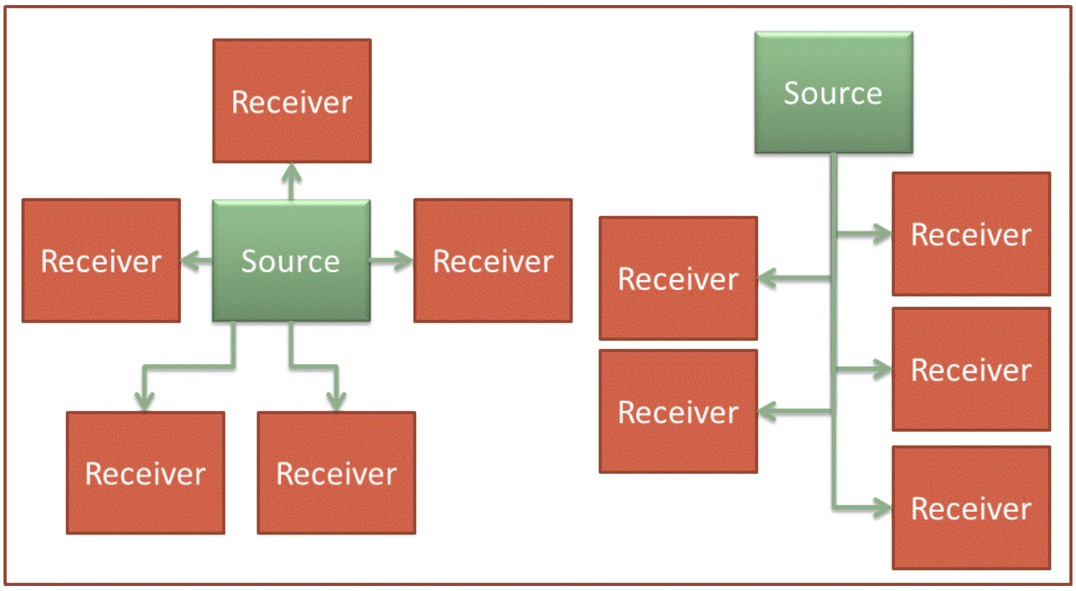


Figure 31: ARINC 429 topologies: star (on left side), bus-drop (on right side).

The Source / Receiver role is contextual to each bus. Indeed each apparatus can participate to several ARINC 429 buses with different roles for each of them.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Few characteristics make ARINC 429 a very robust communication bus:

- Bipolar with return to zero data encoding (Figure 32) that avoid physical layer failures of the type “impossible transition of information”;

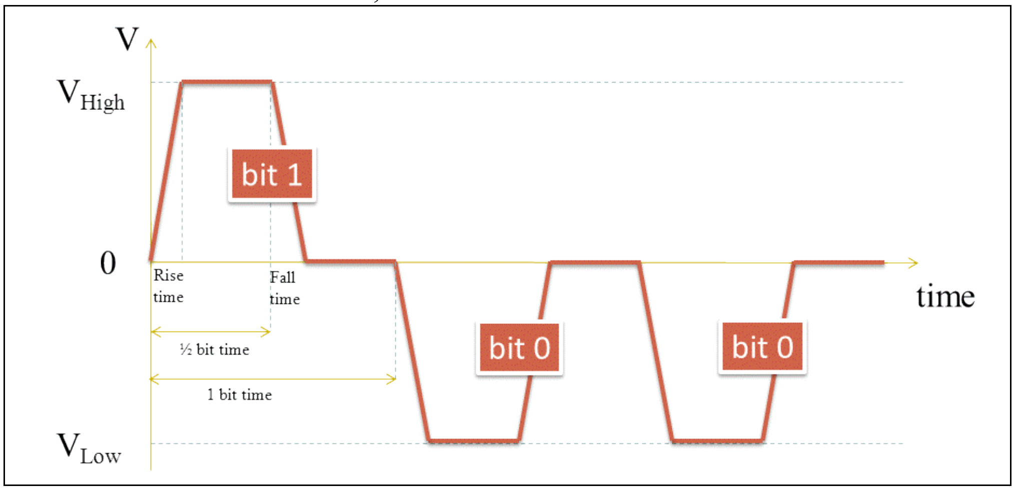
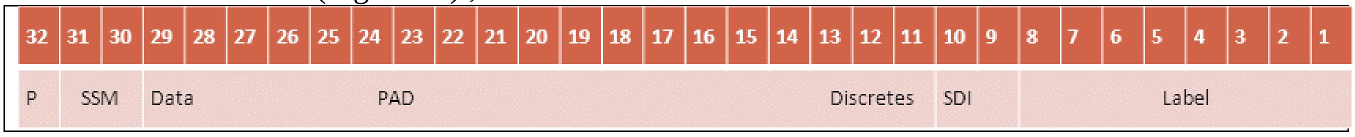


Figure 32: ARINC 429 encoding.

- Fix data words of 32 b (Figure 33) ;



32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
P	SSM		Data						PAD											Discretes		SDI						Label			

Figure 33: General ARINC 429 word structure.

  - P: Parity;
  - SSM: Sign / Status Matrix: for particular critical data;
  - Payload;
  - SDI: Source/Destination Identifier;
  - Label: type of data and corresponding identifiers.
- Emitter signature in SDI.

Next section presents the failure modes associated to this communication interface and subchapter 8.2.4 details corresponding causes on COTS blocks.

7.4.2 Failure modes

7.4.2.1 Loss of message

Loss of a frame in A429 can be causes by the media, the emitter or the receiver.

A failure of the media, for instance an open or a short circuit will cause an impossibility to transmit a frame.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

A failure on the emitter ARINC 429 Driver can be of different types. Frame may not be emitted or be emitted in a format that cannot be decoded by the receiver or be emitted to the wrong address so that the normal receiver will never receive it.

A failure on the receiver can cause the loss of the message and more generally of the information it carries when:

- it never transmits the received message to higher layers;
- it corrupts the information so that it is no longer interpretable.

7.4.2.2 Untimely transfer of message

Untimely transfer of message can be generated by emitters or receivers for instance in case of saturation of some FIFO.

7.4.2.3 Untimely or forbidden transition of information

As each communication information can be corrupted in such a way that information that should remain stable suffers an untimely transition to another state valid or forbidden transition to a non-valid state. In the case of A429 the particular encoding of information presented on Figure 32 efficiently protect against non-detection of such a failure during transmission and encoding/decoding by physical layers of emitter or receiver.

Such an error can indeed be only generated by higher layers that could corrupt information before its physical encoding or after the decoding.

Another possible corruption of information during its creation, transfer or reception is a forbidden transition of label to another label. Two cases are possible:

- Firstly, the new label does not exist. In this case the receiver will ignore the word receive and the information will be lost;
- Secondly, the new label exists in the receiver label list. In this case the receiver will consider the word considering it comes from another emitter creating impersonation.

7.4.2.4 Impossible transition of an information

The scenarios applicable to “*Untimely or forbidden transition of information”* of information are applicable to this case without modifications.

7.4.3 Intrinsic robustness of the physical layer

- ARINC 429 differential communication line relies on RZ (Return to Zero) bit encoding;
- Voltage and timing are defined for each bit states;
- A Minimum gap of at least 4 bit times is expected;
- On the physical layer, the ARINC 429 bit encoding uses RZ bipolar modulation and is defined with 3 states of a differential signal: Null state, HI for high and LO for low with timely defined durations and transitions;

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- There is no overlap of the voltage levels definition used to define the states. Furthermore the duration of each state is part of the logical state definition as one logical bit time is defined by a state HI or LO half bit time plus a Null half bit time.

7.4.4 Intrinsic failure mitigation mechanisms

Several mitigation mechanisms are embedded in the ARINC 429 message at logical and functional level allowing the possible detection of corrupted data when it occurs:

- Configurable Parity Bit encoding and check;
- Sign/Status Matrix (SSM) bit field;
- Source/Destination Identifier bit field;
- ICD predefined label and data format for different emitter types (industry standard).

7.5 MIL-STD-1553

7.5.1 Description

MIL-STD-1553 is a half-duplex communication serial bus. MIL-STD-1553 uses a command/response protocol that enables highly deterministic communication making it ideal for real-time command and control functions, which typically require the transfer of data at a periodic rate (isochronous communication).
A typical MIL-STD-1553B system consists of:

- A redundant MIL-STD-1553B bus:
  The standard dictates that all devices in the system be connected to a redundant *pair* of buses to provide an alternate data path in the event of damage or failure of the primary bus. Bus messages only travel on one bus at a time, determined by the Bus Controller;

- A Bus Controller (BC)
  The Bus controller is the master on the bus. The centralized bus controller allows the scheduling of data transfers with microsecond accuracy and very low jitter (see footnote **Erreur ! Signet non défini.** on page 92). The bus may support several BC, for redundancy purpose, but only one shall be active at one time;

- A set of subsystems with an embedded Remote Terminal (RT):
  A Remote Terminal can be used to provide an interface between the MIL-STD-1553B data bus and an attached subsystem, a bridge between a MIL-STD-1553B bus and another MIL-STD-1553B bus. RTs are slaves of the BC from MIL-STD-1553 bus point of view.
- It may also comprise a Bus Monitor. However, bus monitors are specifically not allowed to take part in data transfers, and are only used to capture or record data for analysis. Alternatively, a BM is used in conjunction with a back-up bus controller

Resulting bus Topology is depicted on Figure 34

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

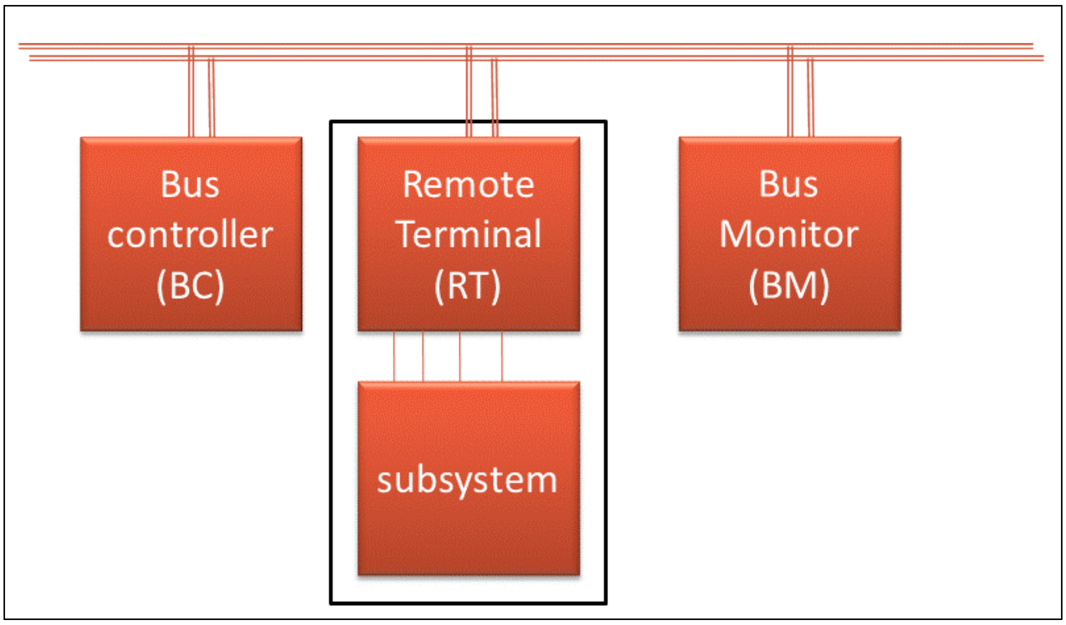


Figure 34: MIL-STD-1553 Bus topology.

The main features of MIL-STD-1553 are

- The connection is
  - Serial,
  - Differential: twisted pair,
  - Isolated: each device is connecter to the main bus by a transformer;
  - Redundant : 2 twisted pairs;
- Data encoding with (Figure 35)
  - Self-clock: the clock pulses are carried by the message itself.
  - Manchester encoding: see Figure 35.

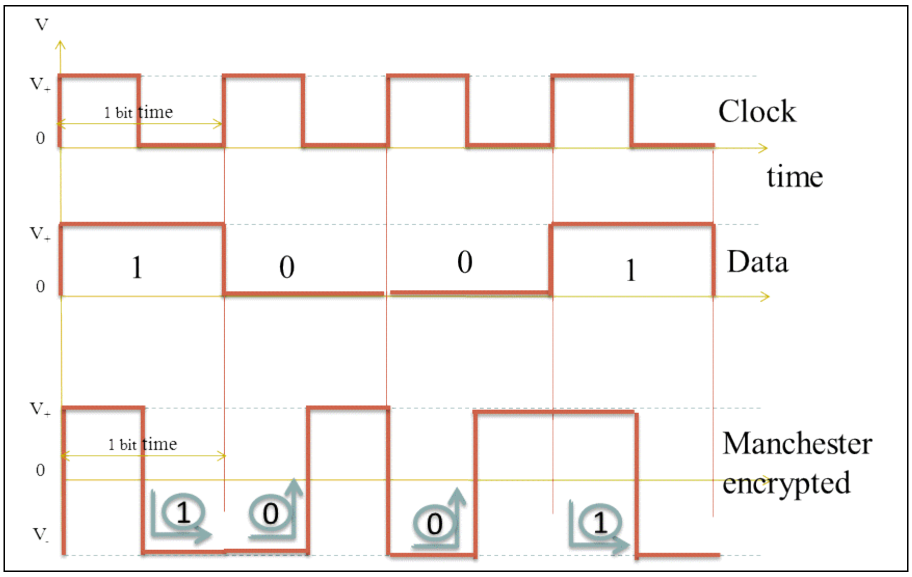
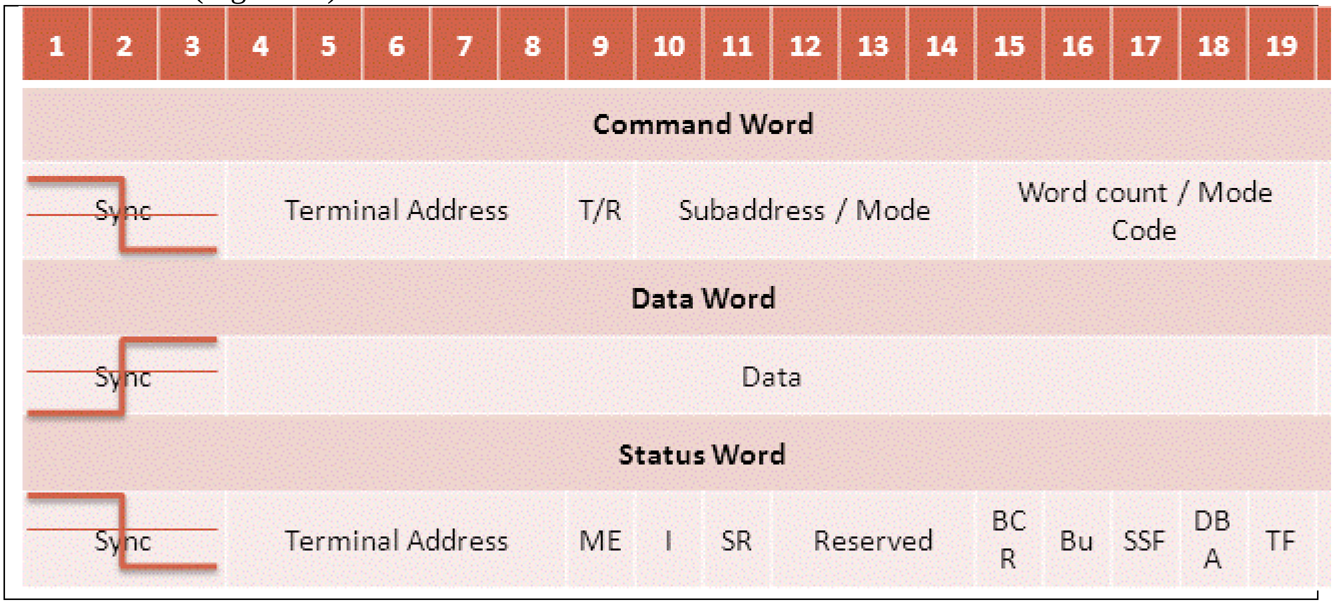


Figure 35: MIL-STD-1553 data encoding;

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Protocol based on a Command-Response approach: The BC Commands, the RT Responds or Acts.
- Word format (Figure 36)



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Command Word																		
Sync			Terminal Address					T/R	Subaddress / Mode					Word count / Mode Code				
Data Word																		
Sync			Data															
Status Word																		
Sync			Terminal Address					ME	I	SR	Reserved			BCR	Bu	SSF	DBA	TF

Figure 36: General MIL-STD-1553 word structure.

Figure 36 presents the three type of word exchanged on MIL-STD-1553 bus.

- Command Word with T/R: Transmit/Received bit;
- Data Word: that effectively carry data;
- Status Word that carry some control and status as :
  - ME: Message Error,
  - I: Instrumentation,
  - SR: Service Request,
  - BCR: Broadcast Command Received,
  - Bu: Busy,
  - SSF: Sub-System Flag,;
  - DBA: Dynamic Bus Acceptance;
  - TF: Terminal Flag.

For all word type the MSB (Most Significant Bit) is the Parity (P) and the three LSB (Low Significant Bit) is the Synchronisation signal[29] (Sync). It is noticeable that "Sync" is identical for command and status words that lead to non-trivial learning phases by Remote Terminals.

- A MIL-STD-1553 messages cannot be reduced to a word. The three types of words combined together form a message. Figure 37 presents a typical sequence message for data transfer between two Remote terminals. It imply the sending of two commands words by the Bus Controller, Data word between the two RT and a status sending to the BC.

[29] The Sync signal duration is 3µs: 1.5µs low followed by 1.5µs high for data words and the opposite for command and status words. Such a sequence cannot occur in the Manchester code. Thus, it allows detecting a word arrival and separate clearly data from other words.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

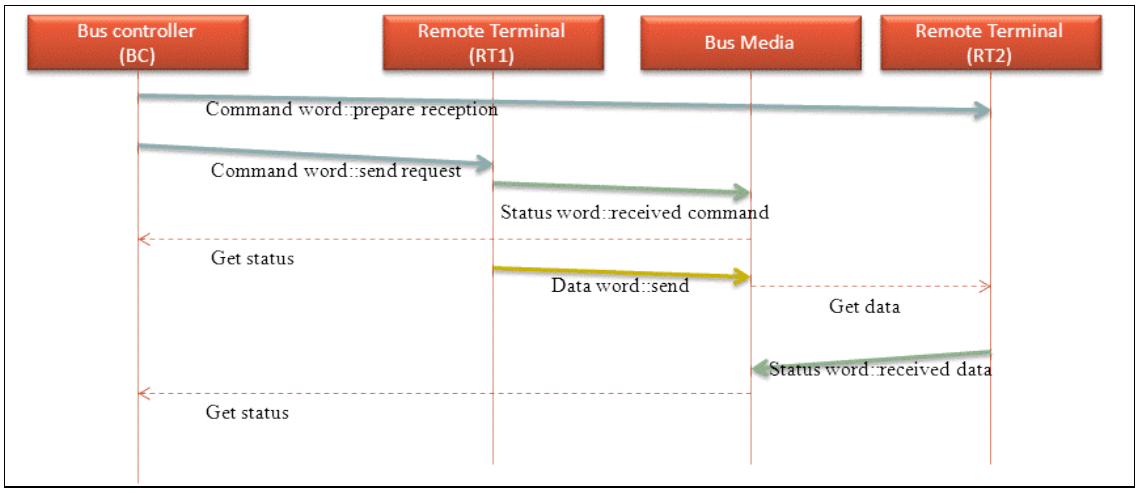


Figure 37: Sequence diagram of a typical message compound of 4 words in MIL-STD-1553.

Many other messages are implemented like for instance control messages with a command word to a RT requesting a status from this RT to the BC.

These salient characteristics of MIL-STD-1553 bus can be classified in first level of OSI (Open Systems Interconnection) abstraction layer model [18]. This classification, performed for instance in [27] in the framework of MIL-STD-1760 [28] helps classifying the failure causes (Table 5).

OSI Layers	MIL-STD-1553 features
Higher layers	Not applicable
Network Layer	Services offered at this level: • Data management: sampling and queuing; • header and identifier, • message sub-addresses, • mode codes, • word count, • T/R flag, • RT status word.
Data Link Layer	• Serialization / deserialization, • Word parity, • "Sync" bits.
Physical Layer	• Manchester encoding, • Tailored sources, • Receiver end characteristics…

Table 5: Mapping of MIL-STD-1553 on OSI layers;

This OSI mapping is not implemented in MIL-STD-1553 that mixes the different layers. It has thus to be considered as guidance for the analysis.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

7.5.2 Failure modes

7.5.2.1 Loss of message

At physical layer, loss of message can be caused by the media, the emitter and the receiver.
In case of an error on both wires of the media (short circuit, open circuit) the signal is totally lost.
In case of an error one of the two wires only one component of the signal can be detected (for instance the V_+). Signal is invalidated at Data Link Layer and considered as lost because the 3 first bits ("Sync" signal on Figure 36) cannot be decoded.
At physical layer, in emitter and transmitter both signals (one on each wire of the twisted pair) are encoded/ decoded separately. The loss of one or both of these two channels will have the same effect

At Data Link Layer, an error in the encoding, transmission or decoding of Sync (transition of sync to a forbidden value – for instance 000 or 111) will cause the rejection of the word and thus its loss.

At network layer, loss of command word will lead to loss of the complete message. Transition of RT address to a forbidden address in command word will also cause the loss of the message.

7.5.2.2 Untimely transfer of messages

At physical layer, a loss of status message could lead to a retry by the BC if a retry process has been defined in layers higher to MIL-STD-1553 layers[30].

At Data Link Layer a Parity error could cause a similar behaviour.

At network layer, in case of corruption of its local RT (Remote Terminal) address, the RT can respond to a BC command addressed to the corrupted RT address and untimely transfer of messages (i.e. impersonation). If no other RT shares the corrupted RT address, the MIL-STD-1553 communication will be effective. If another RT shares the corrupted RT address, a collision will most likely happen between the two RTs. The intrinsic robustness of the communication layer will lead to response interruption after collision detection.

7.5.2.3 Abnormal sequence of message

No cause of abnormal sequence seems to be found in the three layers directly addressed by the MIL-STD-1553 protocol.
Remote Terminal (RT) and Bus Controller (BC) can generate abnormal sequence of messages in their higher layers.

7.5.2.4 Untimely or forbidden transition of information

At physical layer, in case of inadequate shielding of the twisted wired, Electromagnetic field could generate a bit flip and by the way a transition of information to a forbidden state. Indeed transition from an interpreted 0 to 1 necessitate of a V_+ to V_- and V_- to V_+ (see Figure 35).

Error in the Manchester encoder/decoder may also lead to Untimely or forbidden transition of information.

[30] None retry process is defined by MIL-SDT-1553. Such a retry can be useful in case of "on demand messages" and has to be defined in layers higher than MIL-STD-1553 layers.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Data Link Layer has no direct way to create such an *Untimely or forbidden transition of information* transferred. However, an untimely transition of Sync message from command/status to data value may cause its interpretation as the wrong type of message. If interpreted as this new type, the information transferred may correspond to an untimely transition of information.

At network level, untimely transition of address will lead to loss of message for the normal receiver and Untimely transfer of messages for the wrong receiver. This untimely transfer of message may in turn be interpreted by the receiver and constitute an untimely transition of information or a transition to a forbidden state.
If the address is not a valid address, the forbidden transition of address leads to a loss of information.

In case of particular message such as multiple transfer of data word initiated by an unique command word (consider Figure 37 with multiple data words), it is possible that the RT emitters sent different words in abnormal sequence leading at receiver level to an untimely transition of information.

7.5.2.5 Impossible transition of an information

At physical layer, it is possible Manchester encoder could generate such a failure in very particular way (totality or part of words stuck at 0 or 1).

Network and Data Link Layers seem not to be able to generate such an error.

Impossible transition of information can be generated in emitter or receiver at higher layers not directly related to MIL-STD-1553.

Figure 38 presents a summary of the preceding explored failures.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

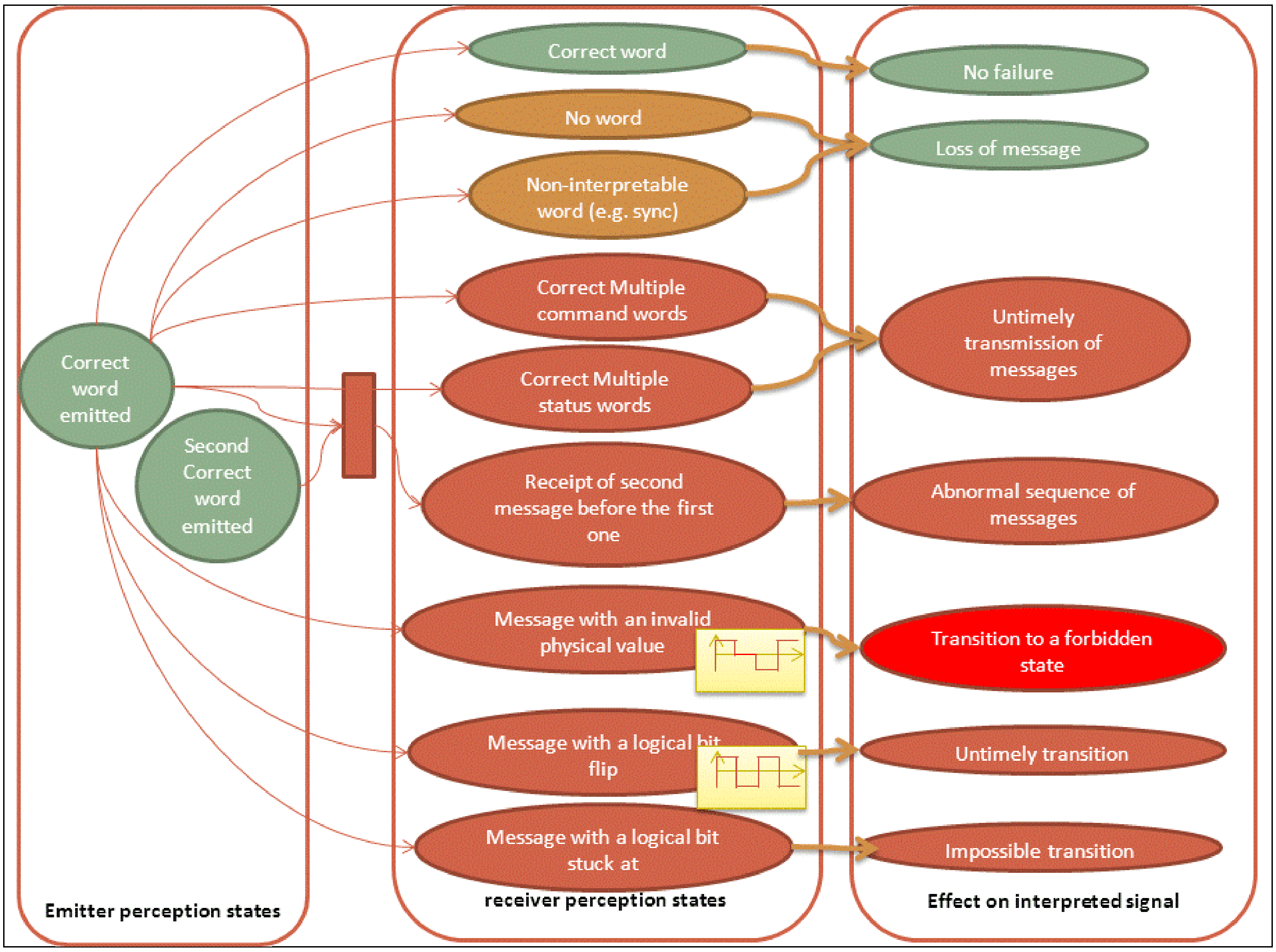

Correct word
No word
Non-interpretable word (e.g. sync)
Correct Multiple command words
Correct Multiple status words
Receipt of second message before the first one
Message with an invalid physical value
Message with a logical bit flip
Message with a logical bit stuck at
No failure
Loss of message
Untimely transmission of messages
Abnormal sequence of messages
Transition to a forbidden state
Untimely transition
Impossible transition
Correct word emitted
Second Correct word emitted
Emitter perception states
receiver perception states
Effect on interpreted signal

Figure 38: Summary of MIL-STD-1553 failures states

7.5.3 Intrinsic robustness of the physical layer

Non Return to Zero (NRZ) (encoding on V_+, V_-) allows detecting most of the word loss situations. Indeed 0v not correspond to an admissible state.

MIL-STD-1553 differential communication line relies on Manchester bit encoding where each transition is timely defined. This protect against most of the causes of impossible transition;

The differential communication line robustness is reinforced, in particular in harsh environmental conditions (lightning strike induced perturbations, induced susceptibility constraints, etc.) by :

- The use of a transformer coupling between the main bus and the device;
- The shielding of the twisted pairs;
- The important ΔV between the states 0 and 1;
- The relatively low bandwidth[31].

[31] The two last points are strongly correlated.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

7.5.4 Intrinsic failure mitigation mechanisms

Several mitigation mechanisms are embedded in the MIL-STD-1553 message at Data Link and Network Layers:

- MIL-STD-1553 is specified redundant (2 pairs of twisted wires);
- Bus Controller centralized and controlled Command / Response protocol;
- Parity Bit encoding and check;
- Predefined command, data or status message format;
- The standardized protocol requires to the RT to only respond to commands received from the bus controller. This response shall be delivered in a certain time slice;
- If a message does not meet the validity requirements defined, then the receiver invalidates the message and discards the data. In addition to reporting status to the bus controller, most remote terminals today are also capable of providing some level of status information to the subsystem regarding the data received;
- RT hardware timeouts will translate babbling idiot temporal failures on the bus to fail silent [27];
- BC and TC monitors the electrical activity of the bus so that emission BC and TC send commands only when the bus is idle since a minimum gap of at least 4 bit times;
- BC can take a sanction by particular command word by making a TC to become silent. In case of redundant configuration of MIL-STD-1553 this inhibition is realized through the second bus.

7.6 PCI BUS

7.6.1 Description

PCI is a parallel bus (32 or 64 bits) embedded on and inter-electronic boards, deployed massively in Personal computer world. It has been initially developed by Intel in 1992 as a microprocessor short distance local bus for interconnection with its resources. Even if its evolutions relax this constraint it is used by many microcontrollers to communicate with their high speed peripherals. On most recent microcontroller it disappears in the benefit of PCIe (serial PCI).

It is a synchronous and parallel bus where address and data are multiplexed on the same lane. It has the particularity to address directly memory zone: word sent are labelled by their address in memory independently of the peripheral. .

Communication Protocol layer

The PCI communication is established amongst a master and a target (slave) under the control of an arbiter. The PCI bus supports multi-master. Indeed, some PCI buses node can be master or target depending on the arbiter contextual arbitration. This arbitration is in general done at applicative level although PCI standard give some recommendations. The arbiter can be included in one of the masters as PCI compatible COTS include usually a PCI arbiter. However, it shall be noted that in general the common PCI arbiters pre-programmed by the COTS manufacturer on a fair arbitration mode (type Round-Robin[32]). This kind of arbitration is not suitable for avionics that constraint access of components to the network. This characteristic imposes introducing an external arbiter (in general a PLD) to master all the transactions and disconnecting COTS master arbiters.

[32] Round-Robin is a scheduling algorithm that assigned time slices to each process in equal portions.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

The structure of messages transiting on a PCI bus is the following (see Figure 39 and Figure 41):

- Address / Data (A/D) that transfers on the same lane addresses and data;
- Command / Byte Enable (C/BE) that transfers the type of action to be accomplished : read, write, etc. ;
- Frame, DEVSEL (Device Selected), IRDY (Initiator Ready), TRDY (Target Ready), STOP that control the device selected for the transaction, the beginning, the end and the request to abort the transaction (see details below);
- REQ (Request) and GNT (Grant) dedicated to the communication between Master type PCI Devices and Arbiter in order for these Masters to request the Arbiter for master role (REQ) and for the Arbiter to authorize them to take this role (GNT).
- CLK (Clock) for synchronisation;
- PERR (Parity Error), SERR (System Error) allow error reporting.

At physical level

- Address / Data is coded on 32 or 64 wires for 32 bits and 64 bits PCI respectively;
- Bus Command / Byte Enable (C/BE) on 4 wires;
- Control information: Frame, DEVSEL, IRDY, TRDY, STOP are coded on 5 wires;

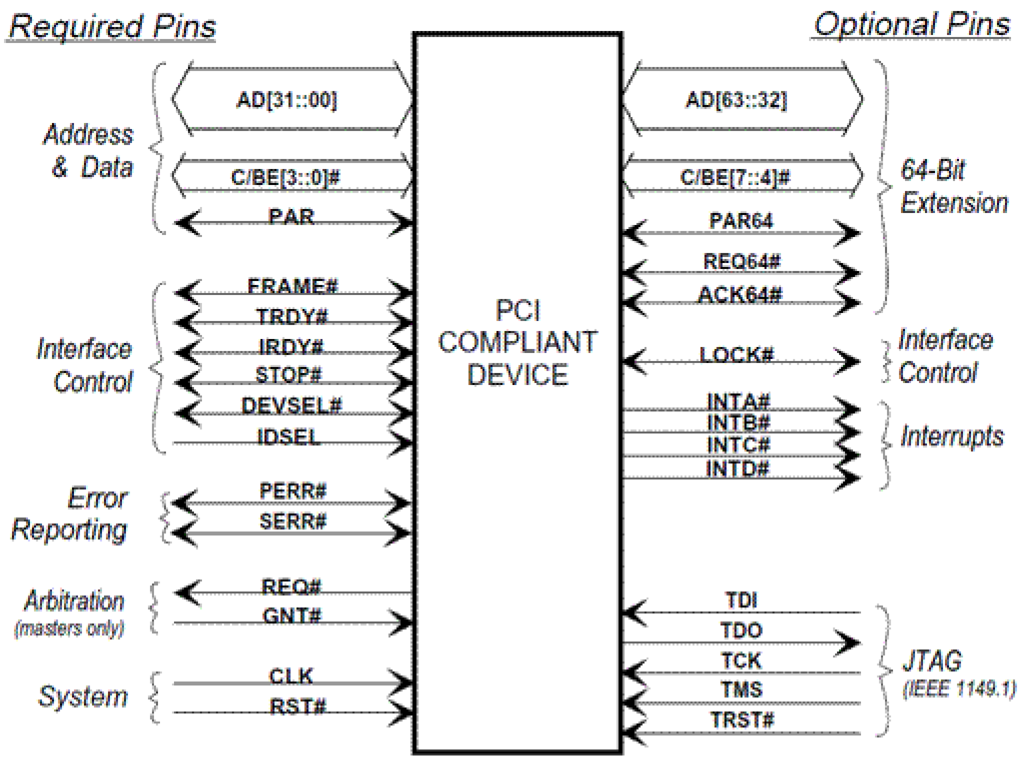


Figure 39 : PCI Pin list from PCI Local Bus Specification [29]

Transfer is requested by a master to the arbiter by pulling REQ to 0V. The authorisation granted by the arbiter to this device to act as master is realized by pulling GNT signal to 0V. Transmission begins by an address phase. During address phase, C/BE indicates if the data transferred will be read or write (others operation are allowed by the standard that are not used in avionics applications). The transmission can be realized by single words or by burst with data sending by packet of several successive words after an initial address sending. During the Data phase, C/BE indicates the number of wires that will transmit the data

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

(Byte enable). Figure 40 shows an example of such a transaction for a read operation. One can note that an address phase (with the appropriate C command) is followed by several data bursts.

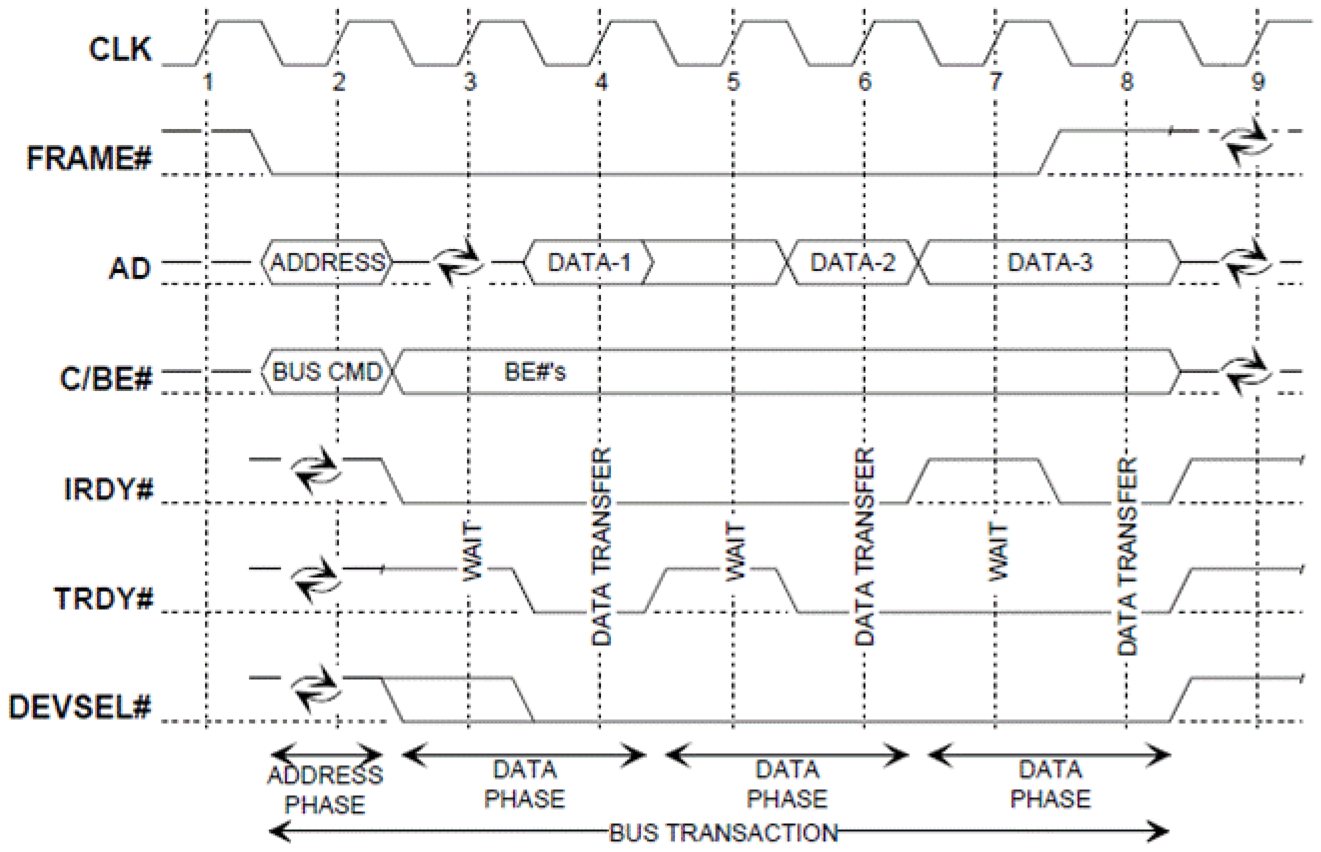


Figure 40: Chronogram for a basic read operation, showing the state of the different signal involved. Signal followed by"#" are active at low voltage level [29]

The master controls the signals IRDY, in order to initiate the transaction, and FRAME in order to control the duration of transaction (Figure 40), in particular for data bursts. On its side, the Target control DEVSEL that allows the target to signal that the address requested by the Master correspond to the one store in its register, TRDY (target Ready) that allows the target to inform that it is ready for transmission and STOP that allows the target to stop the transaction for example if its buffer is full. IRDY and TRDY can pause the transaction; STOP aborts it and liberates the bus for a new transaction. If the STOP signal is set before the beginning of data transfer, the STOP is interpreted by the Master as a Retry request.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

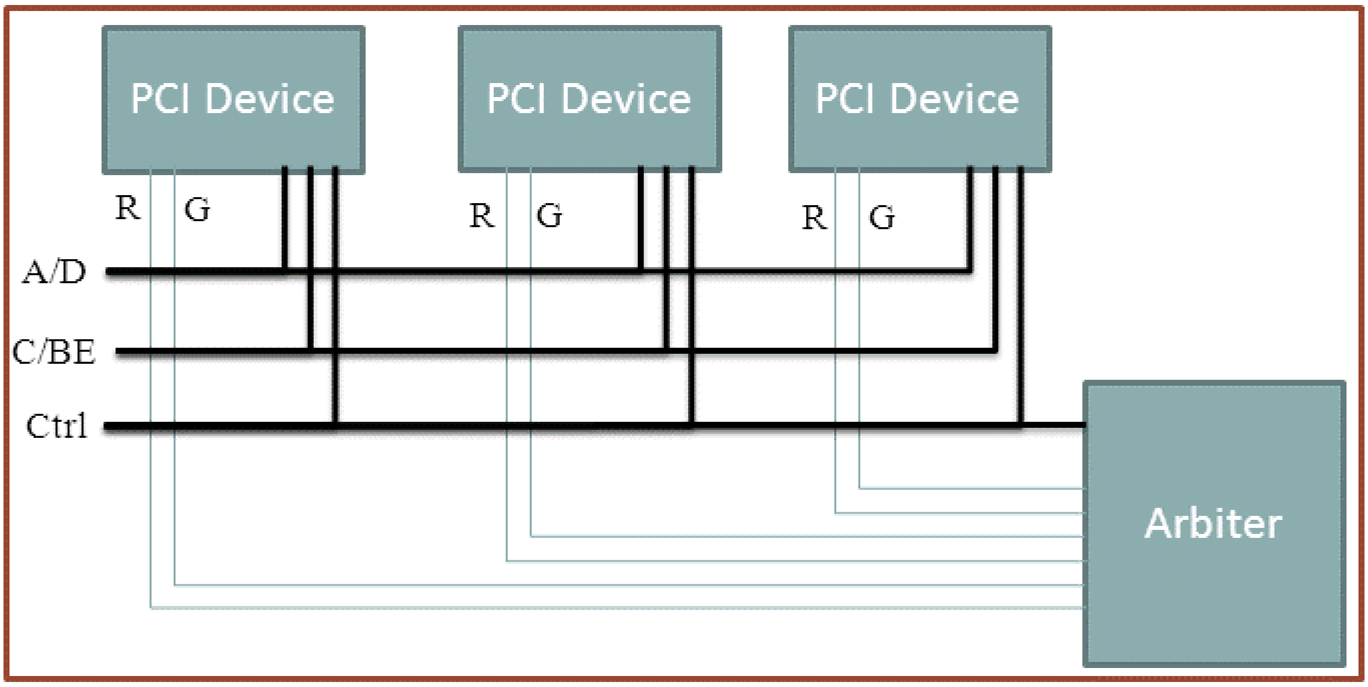


Figure 41: a typical PCI architecture with three connected devices of master type and an arbiter. Only main lines have been represented. Bold line stand for multiple wires; R is REQ and G is GNT.

7.6.2 Failure modes

At the abstraction level addressed in this section a message transmitted on the PCI bus is split in its data, addresses and various controls. The modes loss of message, untimely transfer of message and abnormal transfer of message should be detailed on these different components. In spite of this, we consider that the functional content of the message is contained in its data and Addresses components. Other signals are only considered as enablers of this message transfer. Their failures will then result in failures on data and Addresses components. The clock that can have particular behaviour and impact will be also considered. Interaction between PCI devices and Arbiter will be considered since arbiter plays an important role in the transaction organisation.

7.6.2.1 Loss of message

All the transfers are operated on a TTL type[33] line so that at the basic level of abstraction described no information (of any type) can be lost except the CLK for which a low energy state is not functionally admissible. This CLK lost can be caused by COTS clock loss or PCI interface failure. It can be easily detected by PCI Target on the bus if the vivacity of the bus is monitored[34].
However, considering all the signals except A/D and clock as enabling signals lead to possible loss of Address or Data due to failures in the communication of C/BE or control signals. For instance if IRDY cannot have transition to 1 no address exchange is possible, if it cannot have transition to 0 no data exchange is possible. All these different failures are physical failures of the I/O stages of the PCI devices or protocol errors of the PCI interface block.

[33] « Type » means here that the nominal voltage value is not 5v but 3.3v but the logical principle is identical: low voltage is logical 0, high voltage is logical 1 and intermediate value can be unpredictably interpreted as 0 or 1.
[34] By *Active vivacity* monitoring we mean a monitoring of the network functioning that request common action at both considered terminations, for instance when periodically a Master sent a particular time dependant request to a target in order to monitor that the network is operational. .

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

7.6.2.2 Untimely transfer of messages

a. *Untimely transfer of address*
In general, the abnormal generation of an address transfer is protected by the control signals at PCI interface level. In the case of an address generation by more buried blocks in the COTS, the risk is the untimely transmission of data (see section "Untimely transitions of target address" in the same subchapter.). A particular behaviour could nevertheless appear at PCI interface level. Indeed, if at target PCI interface level, FRAME has encountered a transition to 0 (asserted), then the target will wait until IRDY is settled to 1 (de-asserted). The next transaction initiated by the master can interpreted by this target even if dedicated to another target and can lead to erroneous behaviour:
i. If the C/BE assigns a Read then normal and abnormal targets may emit simultaneously;
ii. If the C/BE assigns a Write the command can be executed in both targets.

b. *Untimely transfer of data flow*
Information transfer between two clocks edges has been treated in the case of SPI. The transfer of data to or from the wrong target or the transfer at the wrong time will be covered in the following paragraphs.

7.6.2.3 Abnormal sequence of message

This mode occurs when PCI interface transfers a data set 1 after a data set 2. These data sets can be sent to or receive from different target or the same target. The PCI interface is simple and is not able to just invert two data sets. Such a failure can only be generated by buried blocks upstream to PCI interface. It is difficult to track but can be partly covered by a process counter.

7.6.2.4 Untimely or forbidden transition of information

a. *Untimely or forbidden transition of target address*
The case of abnormal transformation of data into address has been examined before as a particular case of untimely generation of an address sending. This section examines the transition to an erroneous address.
The failures causes can be located to the PCI interface block or on more buried blocks. Local effect can be a loss of data for the normal receiver and more rarely an erroneous behaviour of the receiver that receive data. Indeed, if error is located on the PCI interface, for instance corruption of an address registers (CONFIG_ADDRESS) this corruption can give non interpretable address. On the other hand, if Address is corrupted upstream of PCI interface block, the wrong address generated can be consistent with existing address. Due to its similarity to normal functioning, such an error cannot be covered by simple ways as already mentioned on SPI errors.

b. *Untimely or forbidden transition of Master address*
PCI can develop such a kind of behaviour close to impersonation. Considering that Master are always ready to initiate transaction (REQ periodically emitted toward the arbiter), if the arbiter untimely grant the wrong Master some data can be lost from a master and data untimely receive

This Document Provided by the Abbott Aerospace
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

from another master that can cause erroneous behaviour of a target if command is Write and of the masters if command is Read.

c. *Untimely or forbidden transition of data*
Transition from a correct data sent or received, to another data. Local effect is an erroneous behaviour of the receiver. Causes are similar to those of *Untimely or forbidden transition of target address*.

7.6.2.5 Impossible transition of an information

a. *Impossible transition of target address*
This mode corresponds to an impossible transition to a correct address. It can be caused by the PCI interface that is stuck on some target address and cannot change it or that cannot generate a particular address. It can be caused by some buried blocks upstream of the PCI interface that request permanently exchange with some target, or cannot generate exchanges with some other. It can also be caused by some electrical error of the PCI interface or downstream elements. It can affect the behaviour of some target that cannot exchange data with a master on time. Like in the case of *Untimely or forbidden transition* it is difficult to separate such a failure from a normal functioning, however a vivacity check can be imagined between some master and some targets (see SPI corresponding section).

b. *Impossible transition of master address.*
This mode is close to the *Untimely or forbidden transition of Master address*. It can be due to the Master itself that become silent on PCI bus in general due to the PCI interface for instance if it cannot request (REQ signal) authorisation to initiate transaction. It can also be caused by an impossible transition of the Grant signal from Arbiter side. Again, such a failure can be close to a functional behaviour and thus is difficult to detect. However, a regular vivacity check can avoid it from being latent.

c. *Impossible transition of data*
This mode corresponds to a data signal remaining unchanged in time. It can be caused by the PCI interface or upstream buried block. The effect of this impossible transition depends upon its duration. Vivacity check, like process counter, can be implemented in order to detect frozen data. If they are computed and decoded by high level software processes they can cover many internal blocks.

7.6.3 Intrinsic failure mitigation mechanisms

- Parity:
  A parity bit is computed by the PCI Device that drives the bus[35] on A/D and C/BE lines and distributed to PCI Device that read the bus through a Parity line (PAR). The PCI Device that read the bus compares the parity it computes with parity distributed on PAR. In case of discrepancy:
    - During the address phase then the target raises a system error (SERR);
    - During the data phase then the target raises a data Parity ERRor (PERR).

[35] This PCI Device is the Master in Master Write operation and the Target in Master Read operation.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE TECHNICAL LIBRARY ABBOTTAEROSPACE.COM

SERR and PERR inform the master about the error. The standard does not specify the backup mode to be applied. For instance in a PC, SERR can conduct to a reboot and PERR to message sending retry.

- Master abort:
  In PCI protocol, if no PCI device has answer to an address request after three clock pulses; the message is decoded by a default device on the fourth pulse. At the fifth pulse if no response has been obtained the master status become "Master abort". This mechanism can help determining that a device is silent on the bus.

It should be noted that these failure mitigation mechanisms cover the communication media and the downstream part of PCI interfaces as they are implemented in these interfaces.

7.7 PCIE BUS

7.7.1 Description

Contrary to PCI that was a parallel communication bus, PCIe is a purely serial bus appeared in 2004 in order to replace progressively PCI. A recent report on the certifiability of this bus in avionics context can be found in [30].
In addition to an extra gain in number of wires, PCIe realize a new increase in bandwidth moreover gain was also obtained in the configurability. Table 6, shows the performances obtained in the 2 variants of PCI (plus PCI-X) and the 6 variants of PCIe.

Specification	Link Width	Link frequency	Max Bandwidth	Transmission	Voltage
PCI	32 bits	33 MHz	133 MBps	Half Duplex	3.3 V (Originally 5 V)
		66 MHz	266 MBps		
	64 bits	33 MHz	266 MBps		
		66 MHz	533 MBps		
PCI-X	64 bits	133 MHz	1066 MBps	Half Duplex	3.3 V
		266 MHz	2133 MBps		
		533 MHz	4266 MBps		
PCIe	1 bit (x1 Link)	2.5 GHz	500 MBps	Full Duplex	0.8 V to 1.2 V
		5 GHz	1 GBps		
		8 GHz	2 GBps		
	x2	2.5 GHz	1 GBps		
		5 GHz	2 GBps		
		8 GHz	4 GBps		
	x4	2.5 GHz	2 GBps		
		5 GHz	4 GBps		
		8 GHz	8 GBps		
	x8	2.5 GHz	4 GBps		
		5 GHz	8 GBps		
		8 GHz	16 GBps		
	x16	2.5 GHz	8 GBps		
		5 GHz	16 GBps		
		8 GHz	32 GBps		

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Specification	Link Width	Link frequency	Max Bandwidth	Transmission	Voltage
	x32	2.5 GHz	16 GBps		
		5 GHz	32 GBps		
		8 GHz	64 GBps		

Table 6: PCI (inc. PCI-X improvment) and PCIe characteristics in GHz, MBps and Voltage [31];

These achievements are due to the use of a serial bus.
Indeed in a parallel bus like PCI routing the board with 32 or 64 tracks induces skew[36] and potential jitter[37] between the different tracks and the clock track. The constraint of synchronisation of signals requests that the clock period is larger than jitter plus skew times[38]. This lower bound on clock period induces an upper bound on the frequency of the parallel bus.

PCIe is a full duplex serial bus:

- Duplex: contrary to PCI that is half duplex because the structure of the bus allows an emission / reception at the same time contrary to PCI in which authorisation to emit has to be requested to an arbiter;
- Serial: data packet are sent on a lane[39] are constituted through successive encapsulation of data acquired in passage throughout abstraction layers.

PCIe communication is structures in 3 abstraction layers (Figure 42):

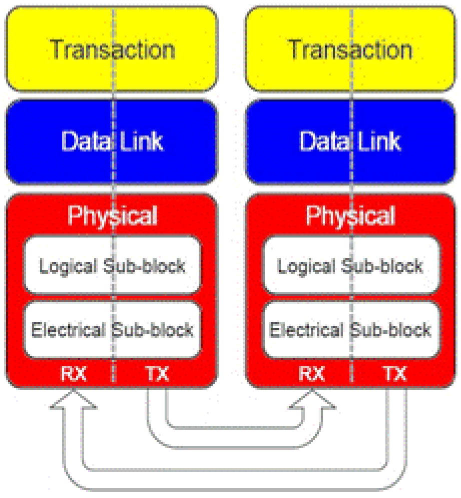


Figure 42: PCIe abstraction layers [32] [31]

- Transaction Layer (TL) ensures:

[36] Skew is the reception time delay between two signals emitted on two different tracks.
[37] Jitter is the undesired deviation from true periodicity of an assumed periodic signal in electronics [8].
[38] The complete equation includes terms related to characteristic times in emitter and receiver. These times rely to silicon technology and decrease with technologies
[39] In more elaborated version of PCIe, serialized data are spread and sent over multiple lanes.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

- Flow Control Management: Flow Control is a Credits based mechanism (see section 7.7.4), which objectives are
  - to regulate the transmissions in order to prevent PCIe Devices from Receive Buffer overflow and
  - to apply transactions' ordering rules.
- Assembly/Disassembly of TL Packets

- Data Link Layer (DLL) ensures:
  - Data error detection through the encoding/decoding of a CRC (see section 7.7.4) on data and addresses– see the paragraph "Intrinsic failure mitigation mechanisms" below for details;
  - Retry management (see section 7.7.4) that defines a procedure to retry transactions under certain conditions in particular when no acknowledgement of receipt is receive from the receiver –see the paragraph "Intrinsic failure mitigation mechanisms" below for details;
  - Link state management that manages the power consumed by the PCI devices.
- Physical Layer that ensures:
  - Link initialization, training, maintenance and recovery in order to recognize a PCI device in the bus. This feature of PCIe avoids using of high level resources in order to map the bus;
  - SerDes that serialize and un-serialize packets.
  - Guarantying of maximum values for Jitter and BER (Bit Error Rate) with respect to budget allocated– see the paragraph "Intrinsic failure mitigation mechanisms" below for details;

Each of these features contributes to the packet structure represented on Figure 43.

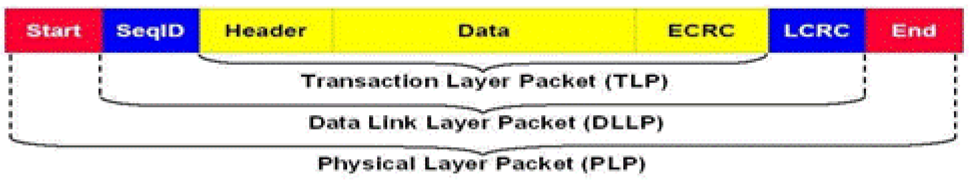

Start
SeqID
Header
Data
ECRC
LCRC
End
Transaction Layer Packet (TLP)
Data Link Layer Packet (DLLP)
Physical Layer Packet (PLP)

Figure 43: Packet structure with the colour code of Figure 42 [31]

The characteristics of the PCIe bus constraints its physical structure. Its serial character structures each lane with a pair of wires. Its character full duplex requests an in-lane and an out-lane (2x2 wires for an I/O-lane). PCIe is implemented using point-to-point links. A set of point-to-point links consisting in a PCIe bus is called a "Fabric" (Figure 44).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

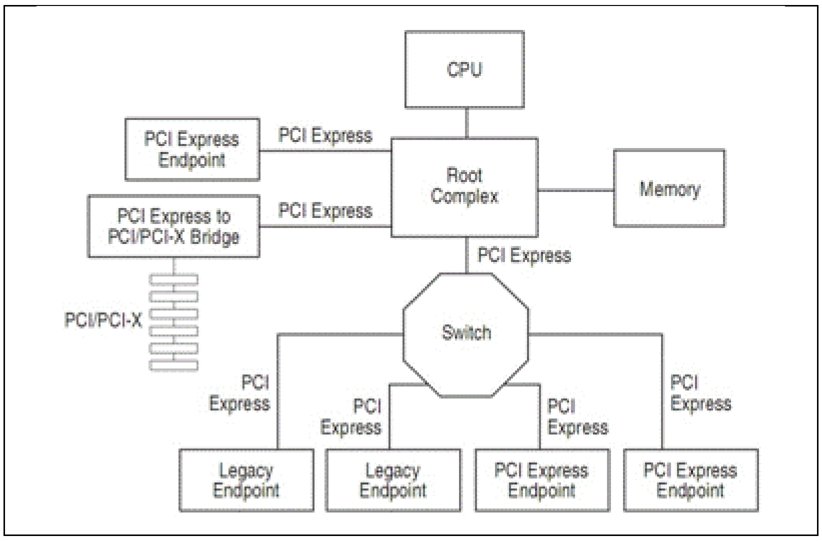


Figure 44: PCIe bus topology [32]

Four different types of components can be found in a fabric:

- A unique root complex that can support several PCIe bus and connect PCIe buses to CPU, Memories, etc. Root complex includes naturally a Switch. The root complex is included as PCIe controller in PCIe Devices like micro-controllers;
- Switch that connects a PCIe upstream branch to downstream branches allowing the connection to several endpoints;
- Bridges that connect PCIe buses to other buses like PCI buses;
- Endpoints represent other PCI devices connected to the bus. Endpoints can be in turn switches that extend the PCI bus in a tree topology.

Considering an endpoint to endpoint communication in the network defined by Figure 44. It is important to note in order to illustrate the importance of the different layers in a safety point of view, that:

- The TLP is encoded by the emission end-point and decoded only by the receiver end-point ;
- The DLLP is encoded and recoded by each PCIe controller crossed along the path.

7.7.2 Failure modes

PCIe structure outlined in the paragraphs above is very complex. Each feature listed in the three abstraction layers has its own state diagram that allows describing its behaviour. The approach already chosen in the PCI failure analysis to concentrate on data and addresses and to consider the other characteristics as address / data enablers will be used again here. In the case of PCIe this choice leads to consider as enablers many packets of information (for instance Acknowledgement packets) or part of the frame like the different CRC as enablers of a correct communication (protocol realisation) and not direct transmission of information to a particular address. Each of the failure modes of the subchapter 6.3 is declined to the three abstraction layers exposed previously and on encapsulated address / data that come from (outgoing packet) or that go to (incoming packet) buried blocks of the COTS. Some of the failure modes explored cannot be generated by some of the application layers, when they are generated by the lower application layers they can be generated by the PCIe controller or some basic features of bus elements, when they are generated by higher layers, they can be generated by buried blocks of considered COTS.

The classified failure modes are of the following types:

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

7.7.2.1 Loss of message

This failure mode can occur at each of the considered abstraction layer through the following mechanisms on incoming or outgoing packets (consider Figure 42 and Figure 43 for analysis):

-a- Corruption of the part of the outgoing packet under the responsibility of the abstraction layer in such a way that the packet cannot be decoded by the corresponding abstraction layer of the receiver;

-b- corruption of the part of the incoming frame under the responsibility of the abstraction layer in such a way that it cannot be acquired by the upstream abstraction layer;

-c- untimely triggering of a failure mitigation mechanism under the responsibility of the abstraction layer on outgoing or incoming frame, in such a way that the frame is blocked;

-d- corruption of the address to which the outgoing frame has to be sent, in such a way that the receiver does not receive the frame.

Applying these schemes at all abstraction layers gives:

a. at physical layer loss of data can be caused simply by ground short circuit or open circuit of the bus for instance or at SerDes module level. Very unlikely SerDes could also corrupt -a- outgoing or -b- incoming frame in a more subtle way that allows it to be sent but not decoded. -c- is not applicable here, as no mechanism can untimely block a frame at physical layer. -d- case corresponds to –b- case in the case of physical layer. In general these faults are quite easy to detect in particular by test when then are due to design errors;

b. at DLL loss of data can be caused by schemes -a- or -b- or by scheme -c- because of untimely detection of a transmission error, for instance decoding error of the LCRC (cyclic redundancy Check) computed on the address and data. The backup mode is preview in this case by the PCIe protocol that asks a transmission retry (Retry Management). Case –d- is unlikely to occur because it should mean that DLL corrupts a part of the frame not under its responsibility. PCIe Device power management by the DLL could also block a transmission. All these errors located on the PCIe controller can be easily covered by tests.

c. at TL loss of data can be cause by schemes -a- or -b- or –c- in the encoding/decoding of the ECRC. A scenario of type -c- associated with Flow Control Management is unlikely to happen as this mechanism contribute to the loss avoidance. The status of type -d- scenario is unlikely as in the DLL case;

d. on more buried blocks (beneath the PCIe controller), -a- and –b- scenario do not correspond to loss of message but to untimely or impossible transition of information (see below), -c- is possible depending of the mechanisms implemented in order to guarantee integrity on these blocks and scenario -d- is of course possible. Like on the other I/O these failures are in general difficult to separate from normal functioning and are difficult to detect. However, in the case of a transaction requested by applicative software and never sent to a PCIe output due to adverse contribution of buried blocks it is possible to imagine redundant way to send the transaction in order to avoid the same ways. For example some transactions can be

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

emitted on an SPI bus in parallel to the emission on the PCIe. Such a redundancy is necessary partial due to the relative performances of both buses.

7.7.2.2 Untimely transfer of messages

Untimely transfer in the sense of babbling cannot be generated on all layers.

a. Physical layer cannot generate such behaviour as it sends directly the frame requested to be sent. An unlikely scenario could generate this behaviour in case of major failure of the Physical Layer that could then send all the frame on the same PCIe link.

b. DLL is a good candidate for untimely transfer of messages because of the retry mechanism that could perform multiple retries of the same frame in case of internal error[40].

   Such a failure can also appear if the physical layer does not transmit received acknowledgment packets to DLL, or if DLL itself does not consider these acknowledgement packets. These scenarios are covered by a maximum retry number set to 4 for a specific frame.

c. TL should not generate any untimely transfer of messages. However it is unable to prevent DLL to do so: as retry management and Process Flow control are handled by two different layers they are transparent to each other. As a consequence, Flow Control mechanism will be unaware if the Data Link retries repeatedly its transmission due to errors on the link.

d. Buried blocks (beneath the PCIe controller) can generate requests to send extra data on the bus depending upon their complexity. This is studied in next chapter.

In the sense of delay in the message transfer, some abstraction layers have not the same impact

a. Physical layer cannot generate such behaviour as it sends directly the frame requested to be sent. It can contribute by a non-acquisition of acknowledgment of receipt so that the frame is sent through retry management;

b. DLL can delayed transfer because it can lost acknowledgment of receipt and then enforced retry mechanism;

c. TL can contribute to delayed frame through Flow Control Management as its objectives are to regulate the transmissions and to apply transactions ordering rules. The possibility of Maximum Execution Time drift has to be considered regarding this mechanism.

d. Buried blocks (beneath the PCIe controller) can generate delays that are studied in next chapter.

[40] A possible cause of error can be linked to discrepancy between Ack_Latency_timer at target level and replay-timer at initiator level. Indeed this two timers regulate the maximum latency time for acknowledgement sending by the target, which can itself occupied by other frame sending, and the maximum time the initiator wait the acknowledgment before to send a new frame. These times rely partly to initial synchronization of PCIe devices.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

7.7.2.3 Abnormal sequence of message

This failure mode can occurs when data are stored in some buffer or memory before sending or resending. It is in particular the case in the TL where the packets are stored in transmit or receive buffers containing several packets waiting for processing and in DLL where they are stored in a retry buffer before in order to be resent if the acknowledgement of receipt is not received.

7.7.2.4 Untimely or forbidden transition of information

This topic corresponds to the transition to a wrong value of data or address values.
Even if PCIe do not modify the informational content of the frame, it stores it in some buffers and transfers it so that some corruption can result.

- In case of *Untimely transition* at physical layer, the LCRC and ECRC (see next paragraphs) will detect the failure unless their encoding/decoding are themselves faulty.
- In case of untimely transition at data link layer, the ECRC will detect the failure unless its encoding/decoding is itself faulty.
- In case of untimely transition at transaction layer, the ECRC may not be effective.
- Buried blocks (at higher abstraction layers than the ones covered by PCIe controller) can of course realise such a transition that is difficult to detect, except by redundant information ways like the one already mentioned in the case of loss of information.
- In case of forbidden transition (for instance a frame too short) at physical, data link or transaction layer, the frame will be lost.

7.7.2.5 Impossible transition of information

The previous comments apply to this case. Ways to covers this scenario differs as it is possible to implement active vivacity mechanisms such as process counters.

7.7.3 Intrinsic robustness of the physical layer

- The PCIe frame is transmitted through two differential tracks. This characteristic improves robustness.
- The PCIe SerDes encode 8bits on 10bits in order to eliminate too long series of stationary bits (00000 or 11111) that may lead to synchronisation loss due to the high frequency of the communication. This mechanism necessary to the correct behaviour of the bus allows also clock recovery.
- The physical layer is configured in order to maintain the Bit Error Rate (BER) under certain limit.

7.7.4 Intrinsic failure mitigation mechanisms

There are 3 types of failure mitigation mechanisms implemented by PCI Express specification:

a. CRC Computation
b. Frame Re-transmission
c. Adjacent Device's Memory Availability

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

7.7.4.1 CRC Computation

As illustrated in Figure 43, PCI Express uses 2 levels of Cyclic Redundancy Check (CRC) in the frame construction:

- Link CRC or LCRC
- End-to-End transaction CRC or ECRC

End-to-End transaction CRC: ECRC

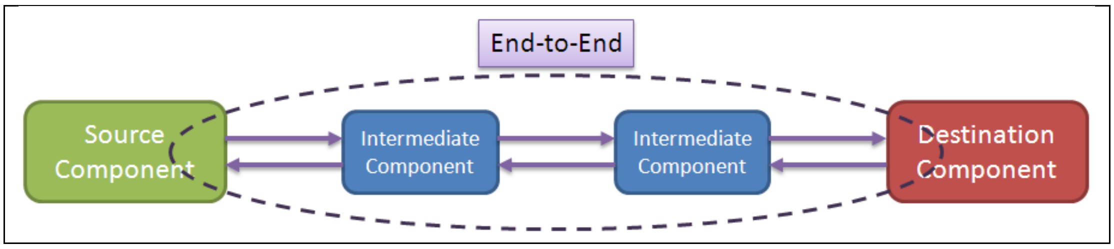


Figure 45: End-to-End transaction CRC

The ECRC is intended to cover end-to-end data integrity. It is the CRC of the constant part of the PCIe frame, i.e the Transaction Layer Packet part of the frame (TLP). The TLP is generated by the source component (Endpoint or Root Complex) and is only forwarded without modification by the intermediate components (such as switches).

As a result, the ECRC might detect a data corruption during intermediate TLP forwarding.

Note 1: The ECRC can be checked by intermediate components if supported and required.
Note 2: ECRC implementation is optional (refer to chapter 2.7 of [32]*)*

Link CRC: LCRC

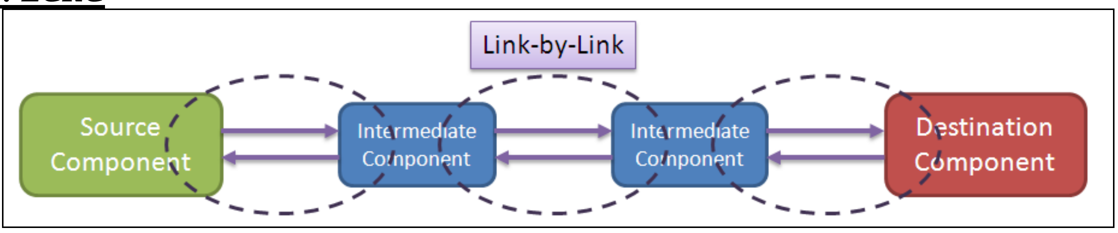


Figure 46: Link CRC

The LCRC is intended to cover Link by Link data integrity. It is the CRC of the Data Link Layer Packet part of the PCIe frame (DLLP). The DLLP is decoded and (re)generated at each step of the end-to-end communication, meaning that every intermediate component manipulates its data.

As a result, the LCRC might detect a data corruption during the frame transmission between 2 adjacent components.
However, data corruption occurring internally to an intermediate component might be masked since the LCRC computation is based on this data.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

7.7.4.2 Frame Re-Transmission

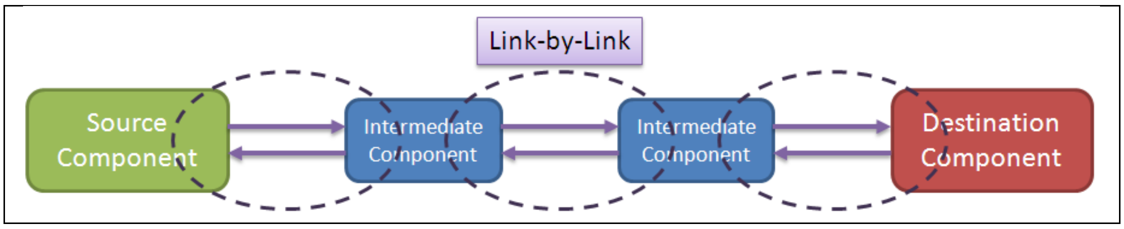


Figure 47: Frame retransmission principle

The frame re-transmission is implemented on a “Link-by-Link” basis in the Data Link Layer of PCI Express and is called “Retry Management”. It consists in frame (TLP) re-transmission (called “retry”) in case of transmission acknowledgement issue.

The acknowledgement consists in a DLLP (called ACK) sent by the receiver to the emitter. It validates the correct reception of the specified frame. In case of reception issue, the DLLP returned to the emitter is a NACK (Non-Acknowledgement).

Note: One ACK DLLP may acknowledge several TLPs. When multiple ACK are scheduled for transmission but not yet transmitted, it is possible to collapse them into a single ACK DLLP. When the emitter receives an ACK for a given TLP, it considers that all the previously sent TLPs are correctly received by the receiver.

A retry is performed by the emitter in the following cases:

- A NACK is returned by the receiver
- No ACK nor NACK is returned by the receiver before the acknowledgement timeout value defined by the PCIe Specification formula ([32], chapter 3.5.2.1), mainly dimensioned by the following criteria:
  - Maximum payload size of the frames
  - PCIe link width implemented
  - Internal reception processing delay (treated as a constant value of 76 ns for 2.5GT/s PCIe)

In the case of retry mechanism activation, 4 retries are attempted. If no result is obtained, the emitter commands a Link re-initialization. If this operation fails, the link is considered down.

7.7.4.3 Adjacent Device’s Memory Availability

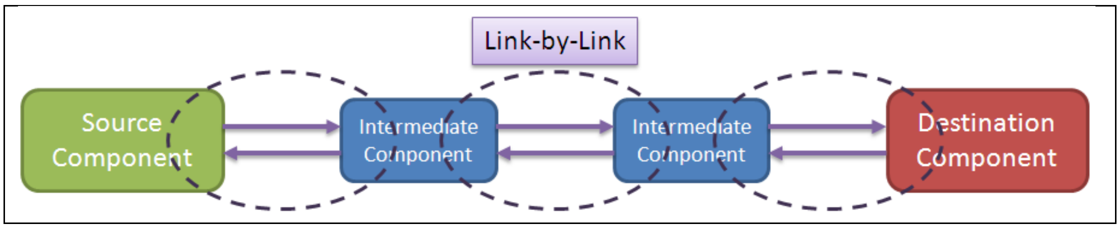


Figure 48: Adjacent device’s memory availability principle

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

PCI Express features a credits based flow control on a Link-by-Link basis. The objective is to anticipate buffers' overflow. Implementing this mechanism, each PCIe component is aware of the load factor of its adjacent components' reception buffers.

The load factor information is contained in DLLP frames called "UpdateFC" (Update Flow Credits), and is transmitted by each component to its adjacent component after a frame reception. As a result, every component is kept informed of the adjacent receiver capacity to accept a new incoming frame, and thus is able to take the decision whether to transmit it or not.

In addition, one type of credit is associated to each type of frame, preventing from irrelevant traffic interruptions.

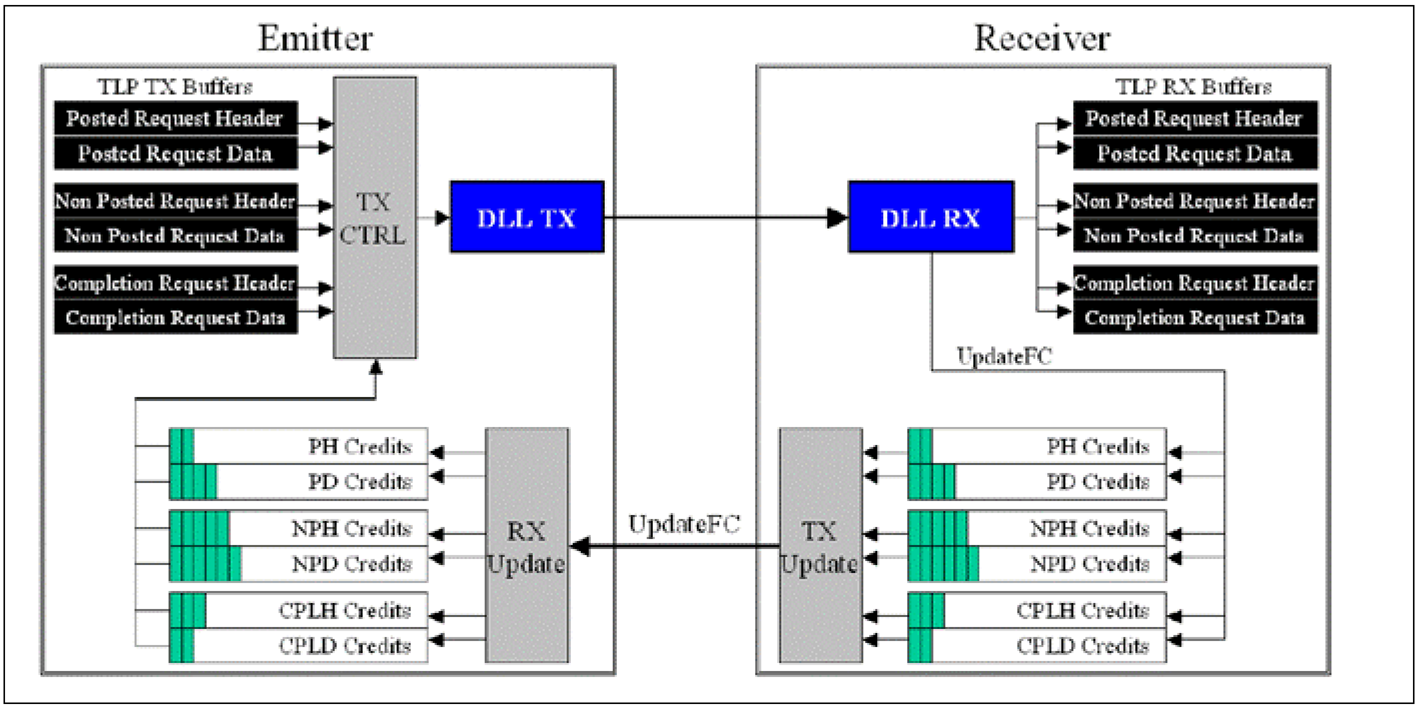


Figure 49: Flow control management buffer overview. Here TLP stands for TL Packet, P for Posted, NP for Non-Posted and CPL for Completion, the suffix H stands for Header and D for Data. [31]

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

8 COTS INTERNAL FAULTS

8.1 INTRODUCTION

The present chapter studies the modes of COTS internal blocks. For each COTS the particular architecture chosen is depicted and block features detailed in order to find their failure modes and to capture block integrated mechanisms designed to detect and/or to mitigate failures. The study performed in Chapter 7 helps separating COTS internal failure mitigation mechanisms from interface failure mitigation mechanisms.
It appears that due to strong interactions of COTS blocks, failure modes of a block can at the end affects many different outputs. These interactions and the associated failures can be revealed at different abstraction levels for the same breakdown level.

Firstly at logical level (see subchapter 6.2), some interactions are documented in the COTS reference manual (see [33] for a particular example). These interactions have to be enriched, still at logical level, by some interactions, which are not publically documented but can be discovered in test or obtained under NDA from the COTS manufacturer. These “Hidden Logical Interactions” are mainly due to some optimisation in the COTS design. For instance, two blocks of interest can have a common access path with a third one (Figure 50).

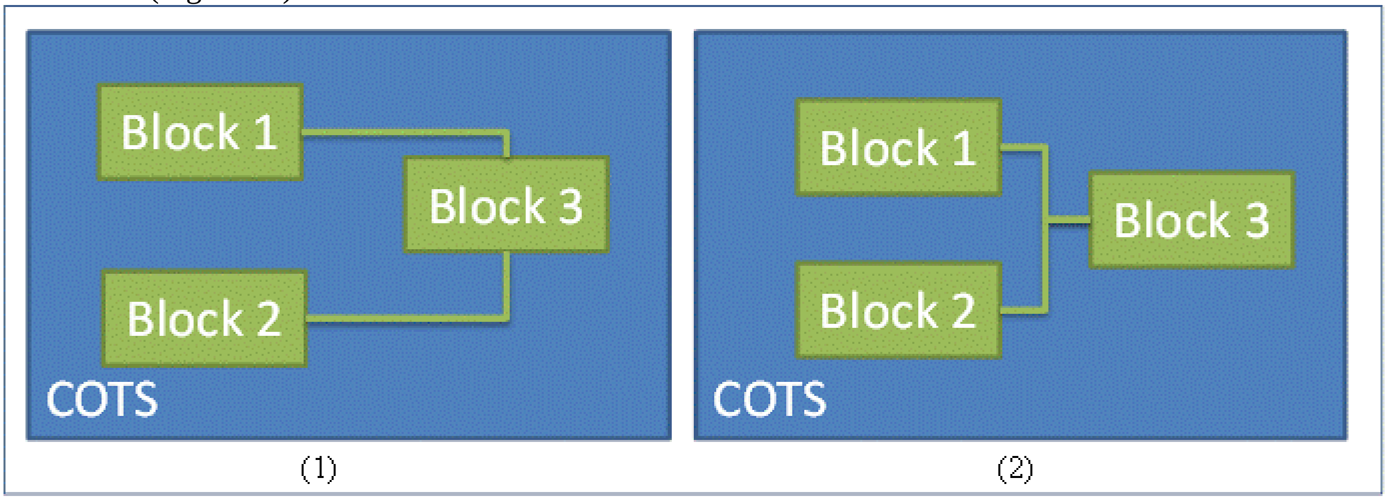


Figure 50: Fictitious example of non-publicly documented common path: (1) Reference Manual representation, (2) Under NDA representation.

From point of view of COTS manufacturers, these Hidden Logical Interactions participate to the difference between a good and a bad design, so that these common paths will be visible on some diagrams obtained under NDA. In the following sub-chapters only failures of documented interactions will be explored.

Secondly, some other interactions can occur at physical level in three cases

- Clock dispatching architecture,
- Power supply architecture,
- Fuse paths[41] (if any).

[41] With the increasing cost of wafer masks, it become less expensive to produce the most complete version of the COTS and to inactivate some internal blocks by fuse in order to obtain a less performing version when needed. Information on fuse is in general difficult to obtain and request in general NDA.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

This physical layer design interactions is studied for the COTS covered by the following chapters on the basis of public document only.
Of the same physical type are some unwanted influences that can be considered as failures. These influences can be due for instance to proximity in the die between some registers and some logical links so that in particular conditions the register content can be influenced by the logical link activity. Such an interaction is not directly addressed in the following subchapter although it can be considered as a cause of some of the possible failures listed. They can be discovered during tests by monitoring key registers (see chapter 9.2.4).

8.2 BRIDGES

8.2.1 Introduction and Available data

As described in section 5.3.1 bridges interface two buses or networks. We consider here a bridge interfacing a PCI bus with a PCIe bus. An example of such a bridge is the Tundra Bridge TSI384. This device connects:

- On one side an up to four lanes PCIe at up to 1GBps per transmit and receive direction and
- On other side a 66MHz PCI or a 133MHz PCI-X.

Addressing is possible in modes: transparent and non-transparent (see section 5.3.1).
Moreover the bridge can be used in two modes with the root (that allocates addressed zones at initialisation) from PCIe side (Root complex) or alternatively from PCI side (host).

The available resources allowing analysing such a bridge are given in Table 7.

Document name	Document description	Examples
User Manual	The user manual describes the feature, the main aspect of architecture, the capabilities and the configuration requirements for the bridge. It explains also how to use the bridge from hardware and software point of views.	IDT® Tsi384 PCIe®-to-PCI Bridge User Manual [34]
Errata list	Device Errata lists the failure reported on the bridge and the particular conditions in which these failures occur. Work around are proposed by the COTS manufacturer and for some failure plan to fix them.	IDT, Tsi384 Device Errata [35]
Application notes	Available application notes should be considered as complementary source of information.	

Table 7: list of input in order to analyse a bridge;

Complementary to these data, it is important to refer to PCI [29] and PCIe [32] standard, in particular the annexes that specify the bridges for two reasons:

- Be confident in the respect of the standard by the component,
- Be aware of the information that is considered as background knowledge by the manufacturer.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.2 Architecture description

Figure 51 presents a typical architecture of a bridge PCI to PCIe. It includes

- A PCIe interface with two channels: Transmission and Reception.
- A PCI interface;
- Buffer management block: Buffers are implemented in the bridge in order to handle the bandwidth and latency differences between the two buses. According to PCIe standard, there shall be several buffers related to the different types of transactions (e.g. non-posted transaction buffer). The buffer management block manage the data dispatching between buffers and the priorities in order to extract them;
- EEPROM controller that interface an EEPROM containing the default configuration;
- Configuration registers;
- Power Management regulating the consumption of the device. These modes are manageable through PCIe requests. Many consumption modes are available ;
- Clocking manages the clock tree. This block can generate clock to other components;
- Reset manages the reset signal. This block can generate reset order to other components;
- Error handling & Interruption manager block:
    - Error handling detects errors;
    - Interrupt manager in manages interruptions from internal block or external devices for internal purpose or propagates interruption to other components.

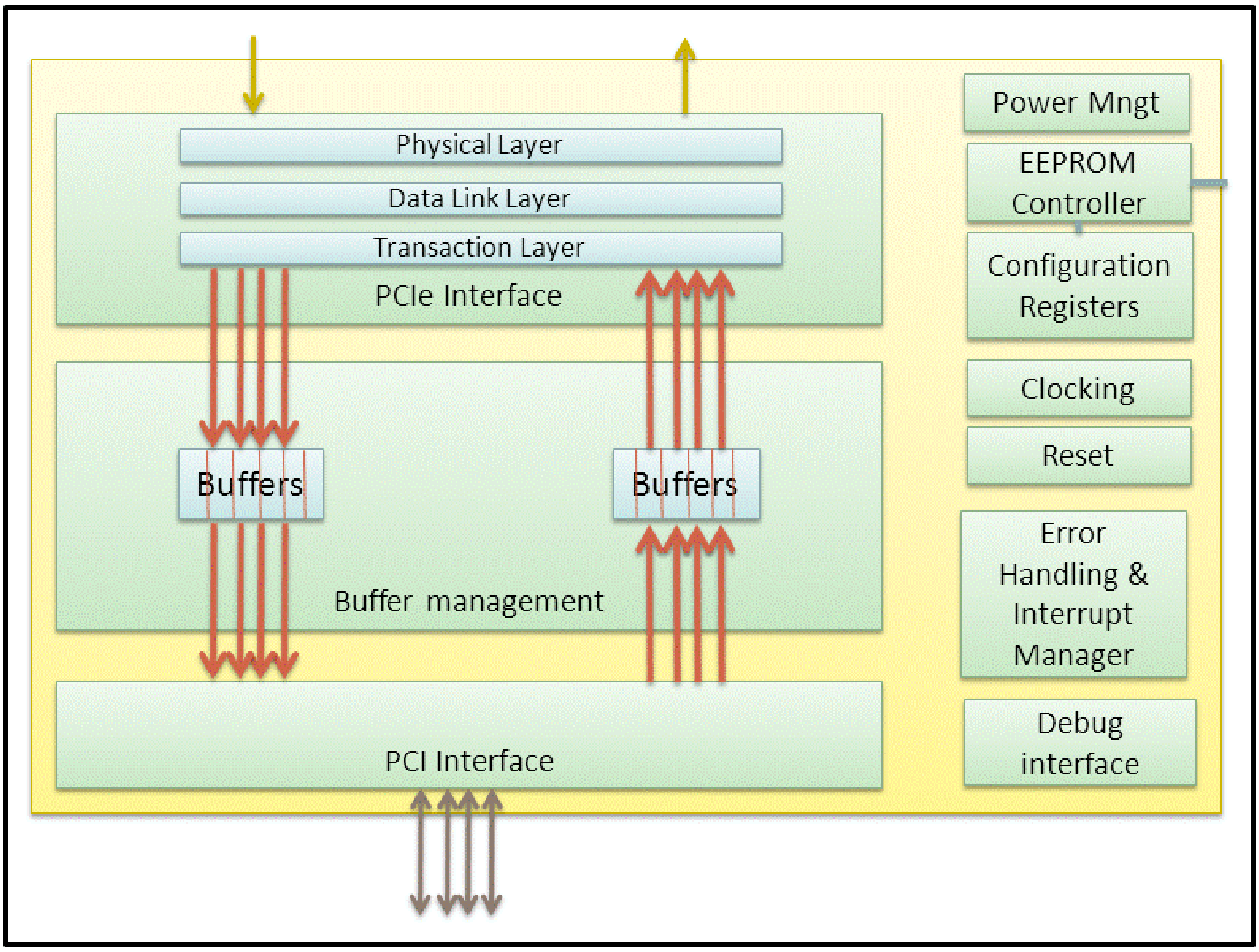


Figure 51: Typical block diagram of a PCI-PCIe bridge [34]).

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

8.2.3 Block study

8.2.3.1 PCIe and PCI interfaces

PCIe and PCI interfaces have been described in chapter 7. Failure modes detailed in the corresponding sections are fully applicable here.

The mixed PCI-PCIe status of the bridge adds nevertheless some behaviour.
As depicted on Figure 52, the bridge is interfaced with PCI device that can be Master or Target and with PCIe device (Root complex or endpoint) that can Transmit or Received transactions.

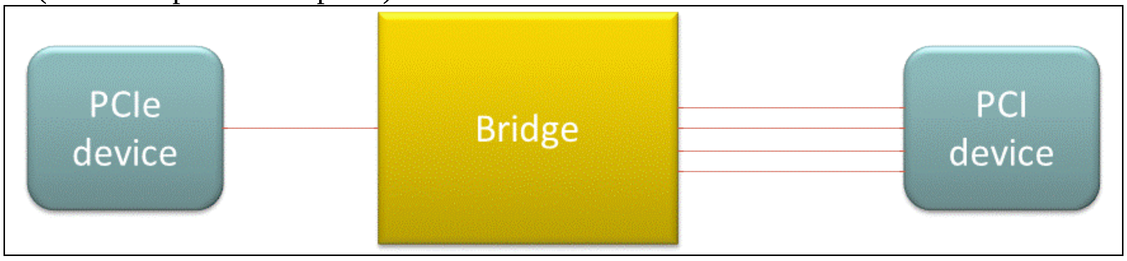


Figure 52: PCI and PCIe interfaces of the bridge.

Loss of PCIe messages: PCIe device will detect that the bridge is silent.

Loss of PCI messages:

If the PCI device is master it will detect that the bridge is silent and will generate a master abort.
If the Bridge is master the transaction can be lost without detection.

8.2.3.2 Buffer management block

8.2.3.2.1 Description

Buffer management block manage:

- PCI read request and corresponding PCIe answer;
- PCIe read request and corresponding PCI data answer;
- PCI write request & data;
- PCIe write request & data.

In conformity with PCIe protocol, buffers include

- Posted transaction buffer for data and some kind of associated headers (buffers on Figure 53);
- Non posted transaction buffer for data and headers (queues on Figure 53).

In fact this block is quite complex and implement several levels of buffer and FIFO that allow storing of transaction before sending on PCI interface and acknowledgement of read completion (central part of Figure 53).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

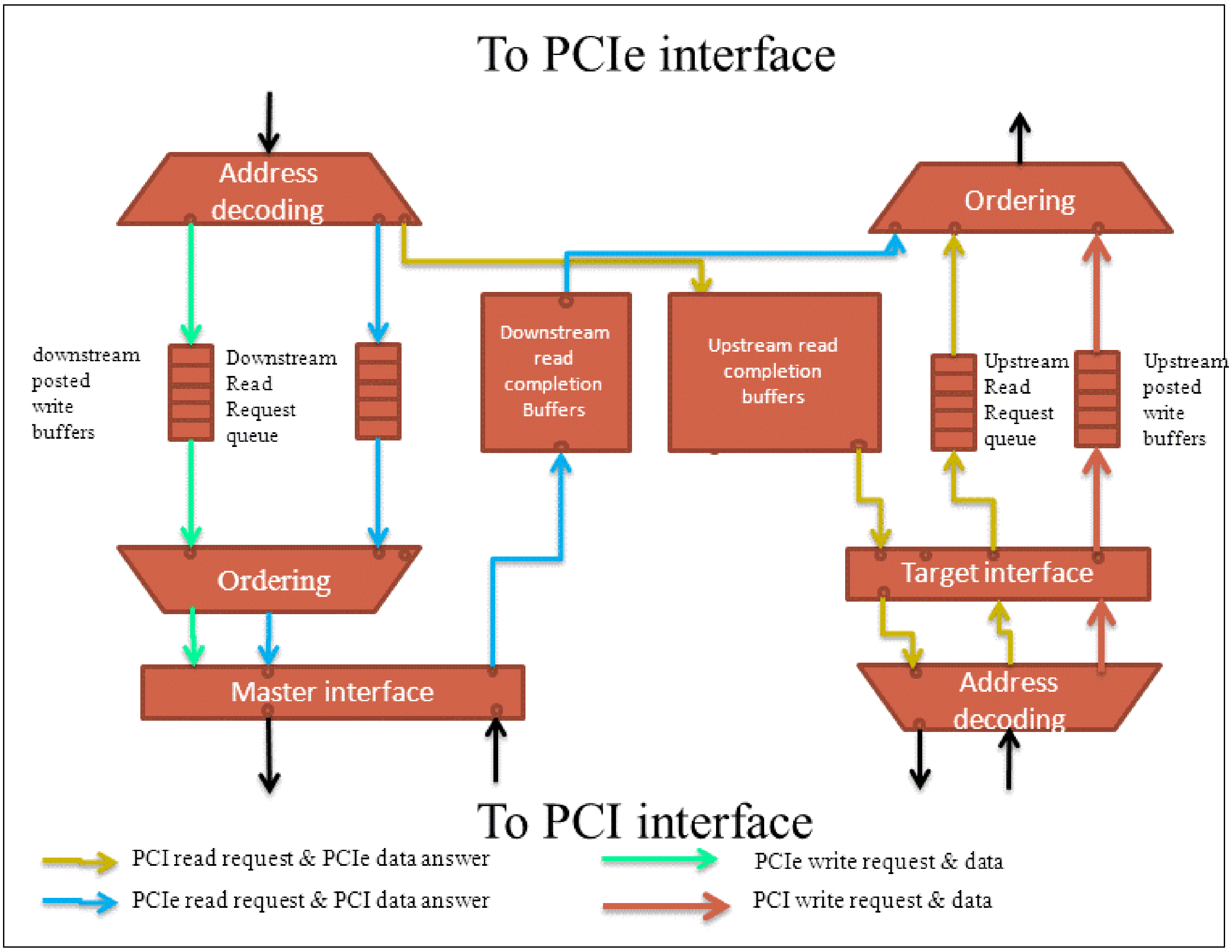

To PCIe interface
Address decoding
Ordering
downstream posted write buffers
Downstream Read Request queue
Downstream read completion Buffers
Upstream read completion buffers
Upstream Read Request queue
Upstream posted write buffers
Ordering
Master interface
Target interface
Address decoding
To PCI interface
PCI read request & PCIe data answer
PCIe read request & PCI data answer
PCIe write request & data
PCI write request & data

Figure 53: Buffer management block structure. (Adapted from [34])

8.2.3.2.2 Failure modes

The major source of failure modes of this block will come from discrepancy between buffers (data) and queue (header). The in/out rate in each buffer frame data shall be the same as the in/out rate in the corresponding header queue. For instance if one queue header is lost, a transaction will not be realized. Reciprocally if a queue header is untimely added, an untimely transaction will be emitted. Following sections detail such kind of behaviours.

8.2.3.2.2.1 Loss of message

The buffer management block can loss transactions typically in case of corruption of a read request queue (header loss). For instance in Figure 54, transaction number 4 and possibly the following will be lost. This behaviour is common in upstream and downstream tracks.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

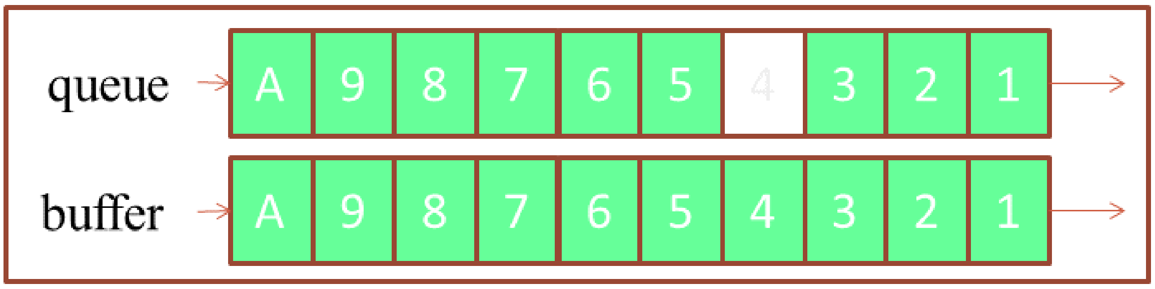


Figure 54: A scenario for transaction loss in bridge;

On Figure 54, data Nr 4 can never be un-stacked from the buffer or may be wrongly processed with transaction Nr 5, and so on.

Note: a missing entry in buffer is described in paragraphs 8.2.3.2.2.4 and 8.2.3.2.2.5.

8.2.3.2.2.2 Untimely transfer of message

A typical scenario for untimely transfer of message occurs if an extra entry is inserted in the queue. In this case when the pointer select this extra entry, an extra transaction is sent with associated undermined data as represented on Figure 55 or a data corresponding to an added waiting transaction as in Figure 56.

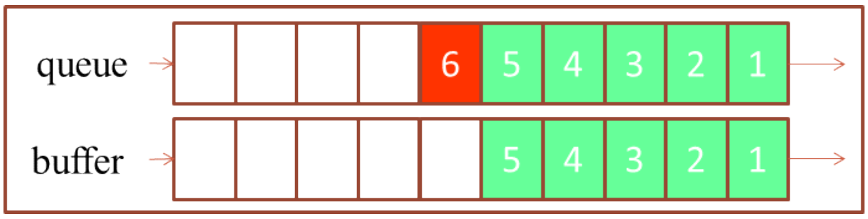


Figure 55: A scenario for untimely transaction emission in bridge. Transaction Nr 6 is send with undetermined data;

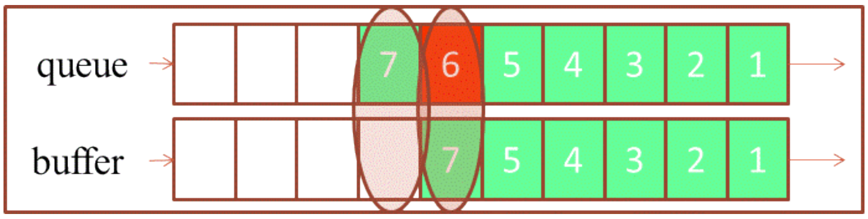


Figure 56: A variant scenario for untimely transaction emission in bridge. Transaction Nr 6 is send with transaction Nr7 data;

Note: Figure 56 shows that in certain conditions this failure continues after the first error occurs.

Others abnormal scenarios may lead to an untimely transfer of message in case of corruption of data in read completion caches and buffers. In this case read completion acknowledgement cannot be sent and in this case command will be retried by the initiator.

8.2.3.2.2.3 Abnormal sequence of messages

In case of buffer manager error at queue filling level, some headers can be inverted so that two messages can be inverted (Figure 57).

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

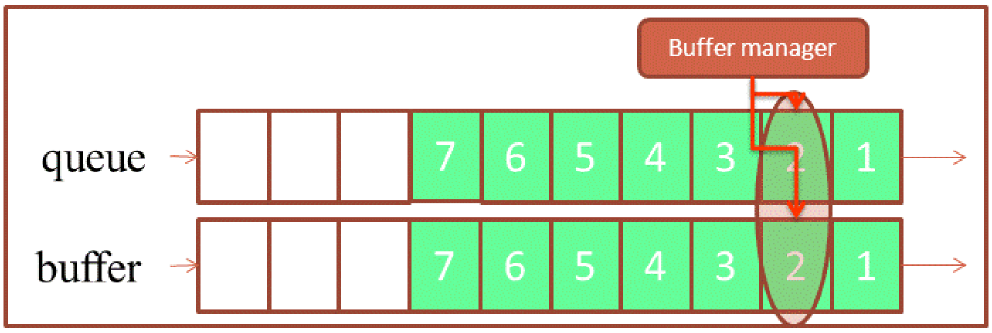


Figure 57: Abnormal sequence due to selection of the wrong entry in buffers (transaction Nr2 will be sent before Nr1).

PCIe specify priority between posted and non-posted transactions. This implies a double structure queue/buffer for posted on one hand and non-posted on other hand, in the buffer management block. This double structure may lead to violation of PCIe priority rules and invert the transmission of a non-posted and of a posted frame.

8.2.3.2.2.4 Untimely or forbidden transition of information

The simplest scenario that can produce untimely transition of information is a data corruption in a buffer (Figure 58).

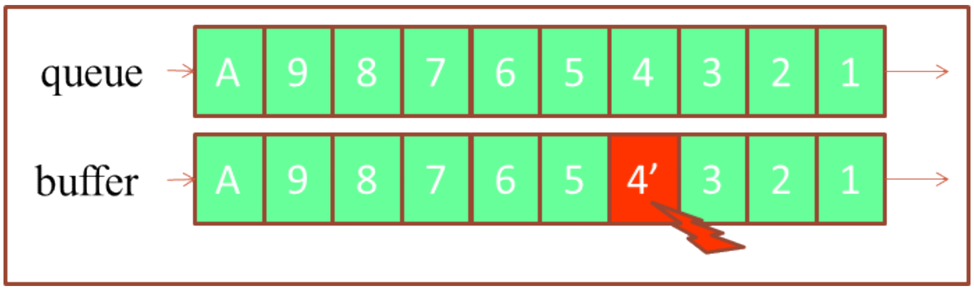


Figure 58: untimely transition of information by direct corruption of data buffer.

Another scenario that could lead to an untimely transition can be caused by buffer manager pointer error so that the pointer on data is desynchronized from the one on addresses (Figure 59).

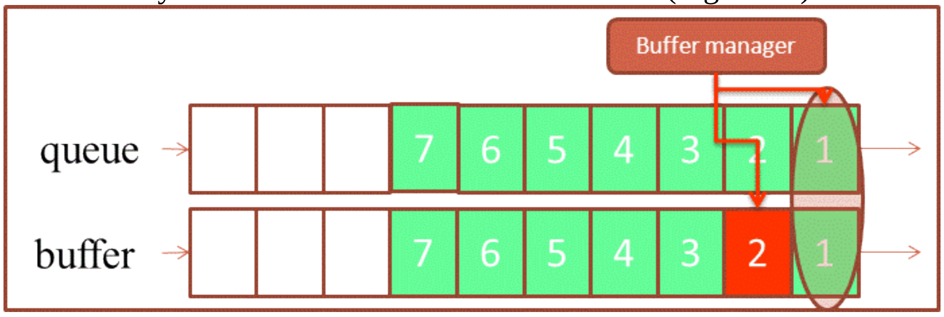


Figure 59 : untimely transition of information by pointer non-synchronization.

This data shift will certainly continue in time so that all data sent will be corrupted.

In case of buffer manager error at address decoding level (see Figure 53), some headers can be inverted so that a message is sent to the wrong address. This can be considered as an untimely transition of address.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Buffer manager pointer errors could also generate a delivery of a posted transaction with data of a non-posted transaction and reciprocally delivery of a non-posted transaction with posted data. Such a failure if it occurs would be permanent because all transaction would be mixed.

Untimely transition of information could be also generated by a pointer location to the wrong queue entry. This could shift the current queue entries by an arbitrary value (Figure 60) and send a transaction with the wrong data.

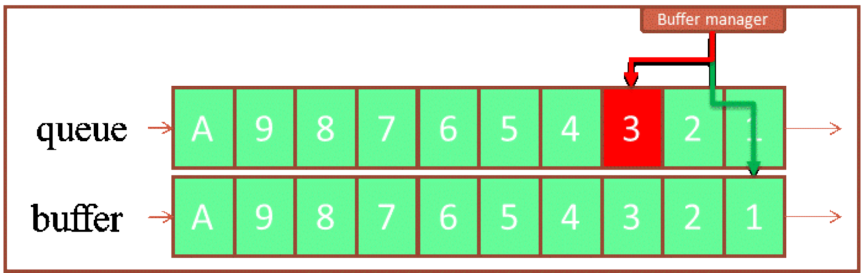


Figure 60: An alternative scenario for transaction loss in bridge;

In case of forbidden transition, the data or the identifiers will not be interpretable and the transaction will be lost.

8.2.3.2.2.5 Impossible transition of information

Most of the failure scenario presented in paragraph 8.2.3.2.2.4 could be apply to an impossible transition of information.
In addition a repeated copy of a transaction in a buffer could stuck an information and lead to this failure mode.

8.2.3.2.3 Failure mitigation mechanisms

It seems that no failure detection allows detecting errors of this block. PCIe LCRC and ECRC are decoded in one direction or encoded in other direction in order to cover errors from the transmission line. These mechanisms are not extended to global mechanisms that could cover the buffers.

Figure 61 shows in a simplified manner the process of error detection based on interfaces protections. In this process, errors detected - for instance by decoding CRC and parity but also all other alert received - are transmitted to Error handling & Interruption manager block for internal sanctions and to the other interface for external information. An error generated at buffer management block will occurs after the protection decoding for incoming flow and before protection encoding for outcoming flow. Such errors will thus not be detected.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

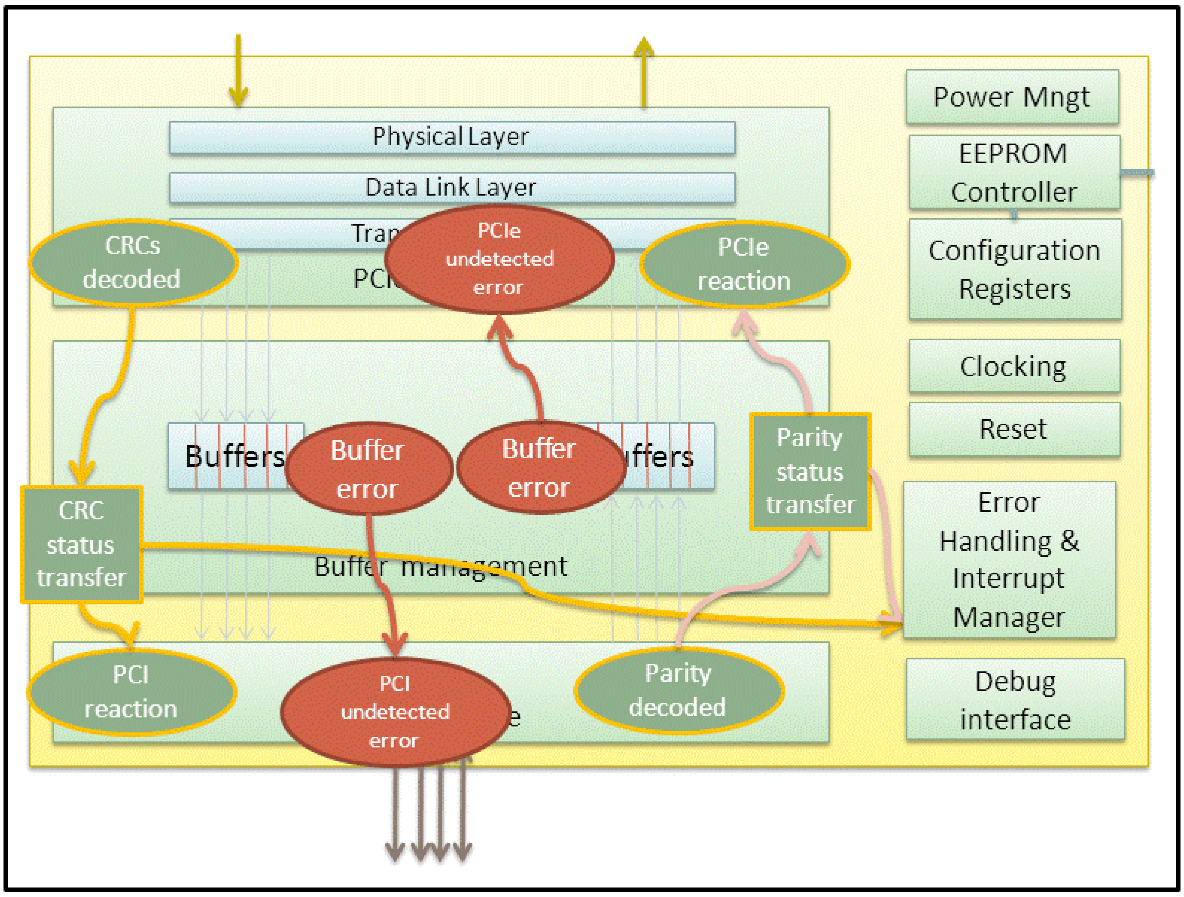


Figure 61: (Non) Detection of buffer management block errors.

8.2.3.3 Configuration and status registers

8.2.3.3.1 Descriptions

Principal functions of configuration registers are to enable and configure the realization of some features (CSR: Configuration Space Register) and to store the status of these features.

The configuration can be charged at power on from an EEPROM through the EEPROM controller or charged dynamically through PCIe bus.

Basic bridge configuration and status are classified in following categories:

- Address Remapping Registers in non-transparent mode,
- PCI and PCIe configuration Registers,
- Advanced Error Reporting Capability Registers,
- PCIe and SerDes Control and Status Registers.

8.2.3.3.2 Failure modes

Due to the principal functions of configuration and status registers, they have to be considered as enablers of transaction transfers, so that their failures will lead to malfunction of transaction transfers or wrong reporting of their status. Two approaches are possible here. The first one considers the failure modes at

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

transaction transfer level and finds the causes on configuration registers. The second considers modes of configuration registers and their effects on transaction transfers. Considering this second option the failure mode of interest are detailed hereafter. In these sections only the modes on configuration are considered. The modes on status are considered as less important.

8.2.3.3.2.1 Loss of message

Applied to configuration register, this mode should be loss of configuration. This is not applicable for a register as it contains always a value.

8.2.3.3.2.2 Untimely transfer of message

Applied to configuration register, this mode is "untimely transfer of configuration". It can occur as configuration can be charged dynamically through PCIe bus. The cause of such a failure is not on the bridge but on the external PCIe root complex that sent a configuration to the bridge. It is very unlikely that the PCIe interface store an admissible configuration sent later to the configuration register.

8.2.3.3.2.3 Abnormal sequence of messages

This mode is not applicable to the configuration register.

8.2.3.3.2.4 Untimely or forbidden transition of information

Applied to configuration register, this mode is "*Untimely or forbidden transition of information of configuration*". It corresponds to an untimely modification of the Bridge configuration. Effects of such a modification depend upon the corrupted configuration zone.
In case of corruption of:

- Address Remapping Registers in non-transparent mode: it will lead to a wrong address mapping and thus to send data to wrong address. This may result in a partitioning breaking.
- PCI and PCIe configuration Registers: It can lead to abnormal functioning of PCIe or PCI interfaces and for instance to loss of messages. It can also lead to send data to wrong address (PCI base address registers) or to generate multiple retry considering the failure mechanism described in subsection 7.7.2.2 and in particular the case signalled in footnote 40 that partly rely on configurations;
- Advanced Error Reporting Capability Registers: Error in this zone can potentially mask errors;
- PCIe and SerDes Control and Status Registers: Error in this zone may lead to failures described in physical layers of PCIe (see section 7.7.2) and in particular to loss of transactions.

8.2.3.3.2.5 Impossible transition of information

Applied to configuration register, this mode is "impossible transition of configuration". It is only relevant in case of configuration change request through PCIe. In this case a PCIe initiated change of configuration should not be effective.

8.2.3.3.3 Intrinsic mechanisms

There is no intrinsic mechanism protecting configuration.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.3.4 EEPROM controller

8.2.3.4.1 Description

Default configuration of the Bridge is stored in an EEPROM controlled at bridge level by a EEPROM controller. At power on the controller charge the default configuration to in bridge internal configuration registers.

8.2.3.4.2 Failure modes

Failure modes of the EEPROM will lead to errors on the bridge internal configuration. Effects of such failures lead to failures listed in sub-section 8.2.3.3 and then on bridge output.

8.2.3.4.2.1 Loss of message

This controller can loss some configuration data so that the corresponding configuration zone is randomly filled.

8.2.3.4.2.2 Untimely transfer of message

Due to some bridge automata error, the EEPROM controller may erase the current bridge configuration by default configuration data during bridge operation. This may have effect only if configuration has been redefined through a PCI or PCIe configuration transaction.

8.2.3.4.2.3 Abnormal sequence of messages

Abnormal sequence in uploading of configuration could lead to errors in several configuration zones.

8.2.3.4.2.4 Untimely or forbidden transition of information

Configuration may suffer of bit flip when transferred by the EEPROM controller.

8.2.3.4.2.5 Impossible transition of information

Configuration may suffer of bit stuck when transferred by the EEPROM controller.

8.2.3.4.3 Intrinsic mechanisms

No intrinsic mechanisms, like checksum, signature, etc., are implemented on configuration uploading from EEPROM.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.3.5 Power management

8.2.3.5.1 Description

Power management block implement the PCI / PCIe power management capabilities mainly used in Personal Computer context for energy saving. In general this feature is deactivated in avionic context.

8.2.3.5.2 Failure modes

Untimely activation of some power management mode may lead in slow down of transaction treatment and so to Maximum Execution Time drift.

8.2.3.5.3 Intrinsic mechanisms

Not applicable.

8.2.3.6 Clocking

8.2.3.6.1 Description

External clock signal is divided on a bridge to signal for PCIe interface, PCI interface in master or slave modes. Clocking architecture rely on several PLL (Phase-locked loop) in order to adapt the 100Mhz reference clock to PCIe clock (2.5GHz) and PCI clock that depend upon the standard applied (PCI or PCI-X – see Table 6). This clocking architecture is represented on Figure 62.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

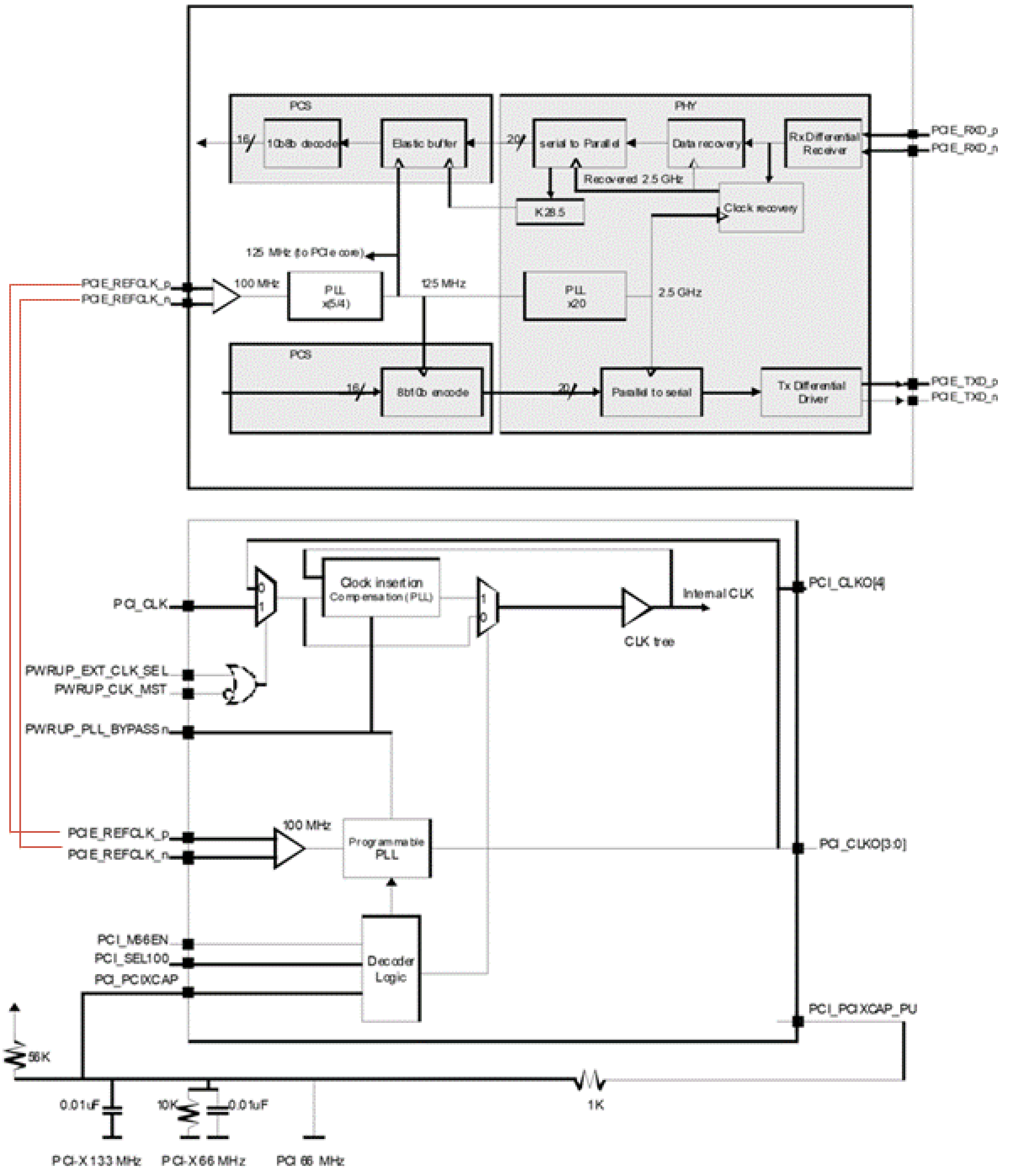


Figure 62

Figure 62 : Clock structure of Bridge TSI384 (adapted from [34]).

8.2.3.6.2 Failure modes

Clocking can be considered as a physical mechanism. The failure mode pattern defined for more functional block can nevertheless applied but adapted at a lower level of abstraction.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE TECHNICAL LIBRARY ABBOTTAEROSPACE.COM

8.2.3.6.2.1 Loss of message

Loss of message applied to clock has to be interpreted as the total lost (i.e. during a time period larger than a few clock pulses) of one of the clocks. Such a loss is in general permanent. The loss of one or of very few clock pulses are described later. Three cases can be considered:

- Loss of the reference clock (PCIE_REFCLK_p or _n on Figure 62). In this case, the complete bridge is silent. The structure exhibit on Figure 62 does not suggest causes for such a scenario;
- Loss of PCIe clock sub block. In this case PCIe interface would be lost (see paragraph 8.2.3.6.2.1). Such a loss could be generated, for instance, by PLL error;
- Loss of PCI clock sub block. In this case PCI interface would be lost (see paragraph 8.2.3.6.2.1). This could be cause by PLL error or by configuration error on the programmable PLL.

8.2.3.6.2.2 Untimely transfer of message

Extra clock pulses could be generated by PLL errors or by PLL configuration error on the PCI clock sub block. De-synchronisation of both block (PCIe and PCI) may cause loss of transactions.

8.2.3.6.2.3 Abnormal sequence of messages

Not applicable as clock signal is periodic.

8.2.3.6.2.4 Untimely or forbidden transition of information

This mode corresponds to extra clock pulses. No mechanisms have been detected that can lead to this failure.

8.2.3.6.2.5 Impossible transition of information

This mode corresponds to loss of clock pulses. No mechanisms have been detected that can lead to this failure.

8.2.3.6.3 Intrinsic mechanisms

None

8.2.3.7 Error handling & Interruption manager

8.2.3.7.1 Description

For sake of simplicity, the two blocks, Error handling on one side and Interruption manager on the other side, have been grouped.

- Error handling block aims at concentrating, ranking and tackling errors by internal means and/or by sending alert to PCIe external devices (see **Figure 63**). As a matter of example, the particular case of PCIe error handling is outlined on Figure 74. The number of errors handled is important (see [34]).
- Interruption Manager: On such a bridge the interruption manager does not generate any interruption. It is dedicated to the transmission of upstream PCI interruption to PCIe and inversely. Contrary to Error handling feature that verify various aspect of incoming messages and take particular sanction, interruption manager only transmit from one bus to the other, particular

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

interruption messages (Message Signalled Interrupt – MSI). For this reason, bridge Interruption Manager is not detailed;

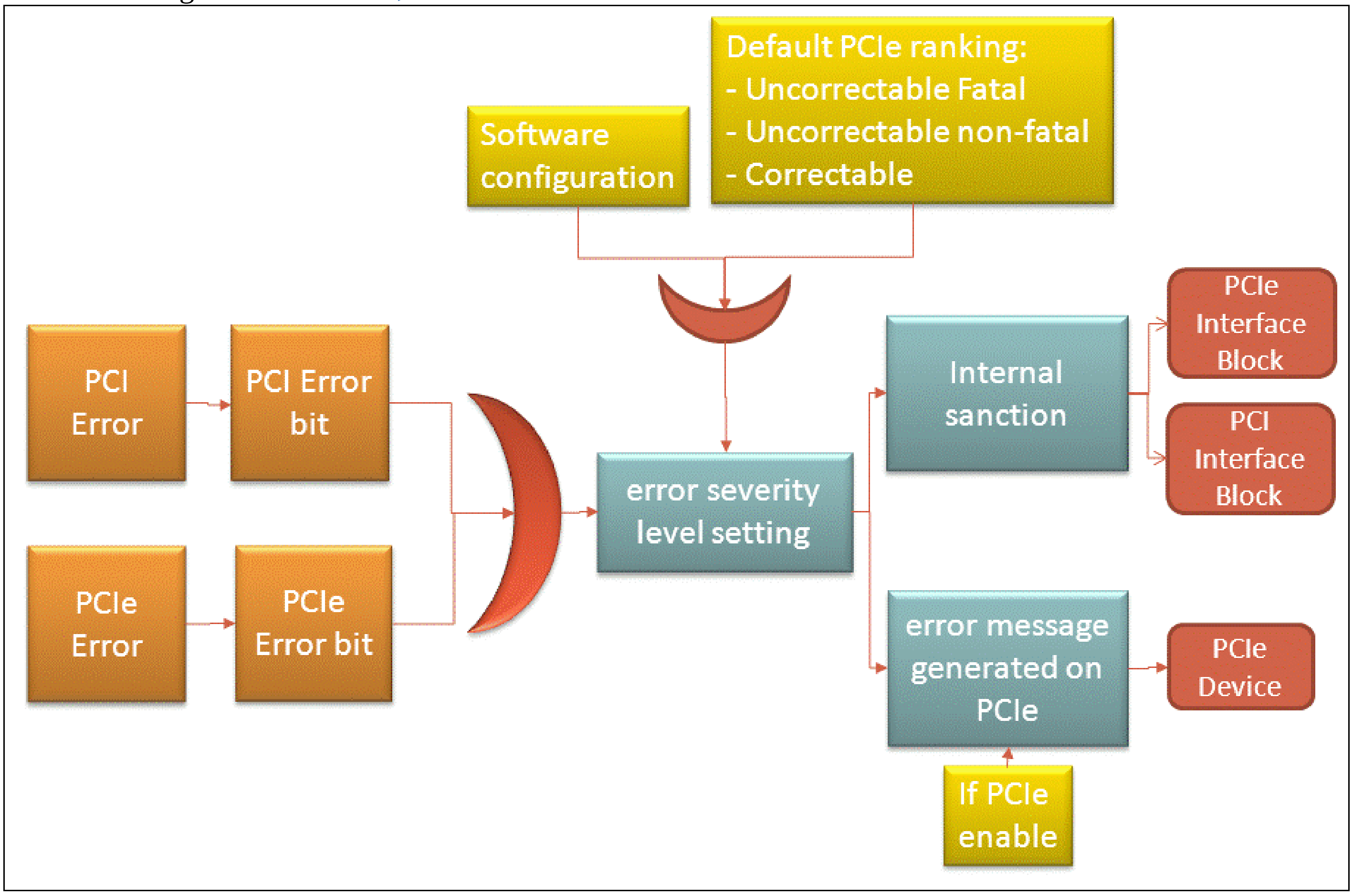


Figure 63 : Bridge error handling principle (interpreted from [34]);

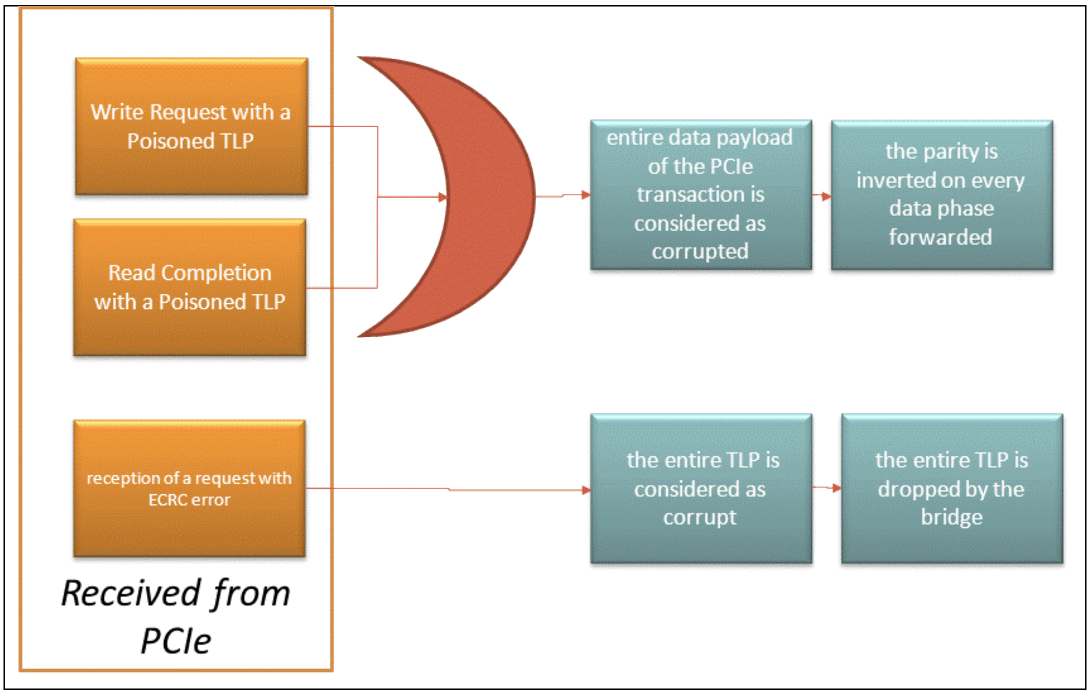

Write Request with a Poisoned TLP
Read Completion with a Poisoned TLP
reception of a request with ECRC error
Received from PCIe
entire data payload of the PCIe transaction is considered as corrupted
the parity is inverted on every data phase forwarded
the entire TLP is considered as corrupt
the entire TLP is dropped by the bridge

Figure 64 : Bridge PCIe error handling principle (interpreted from [34]);

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.3.7.2 Failure modes

Only failures of Error handling block is detailed here on the basis of Figure 63.

Figure 74 summarize the error handling block consequences.

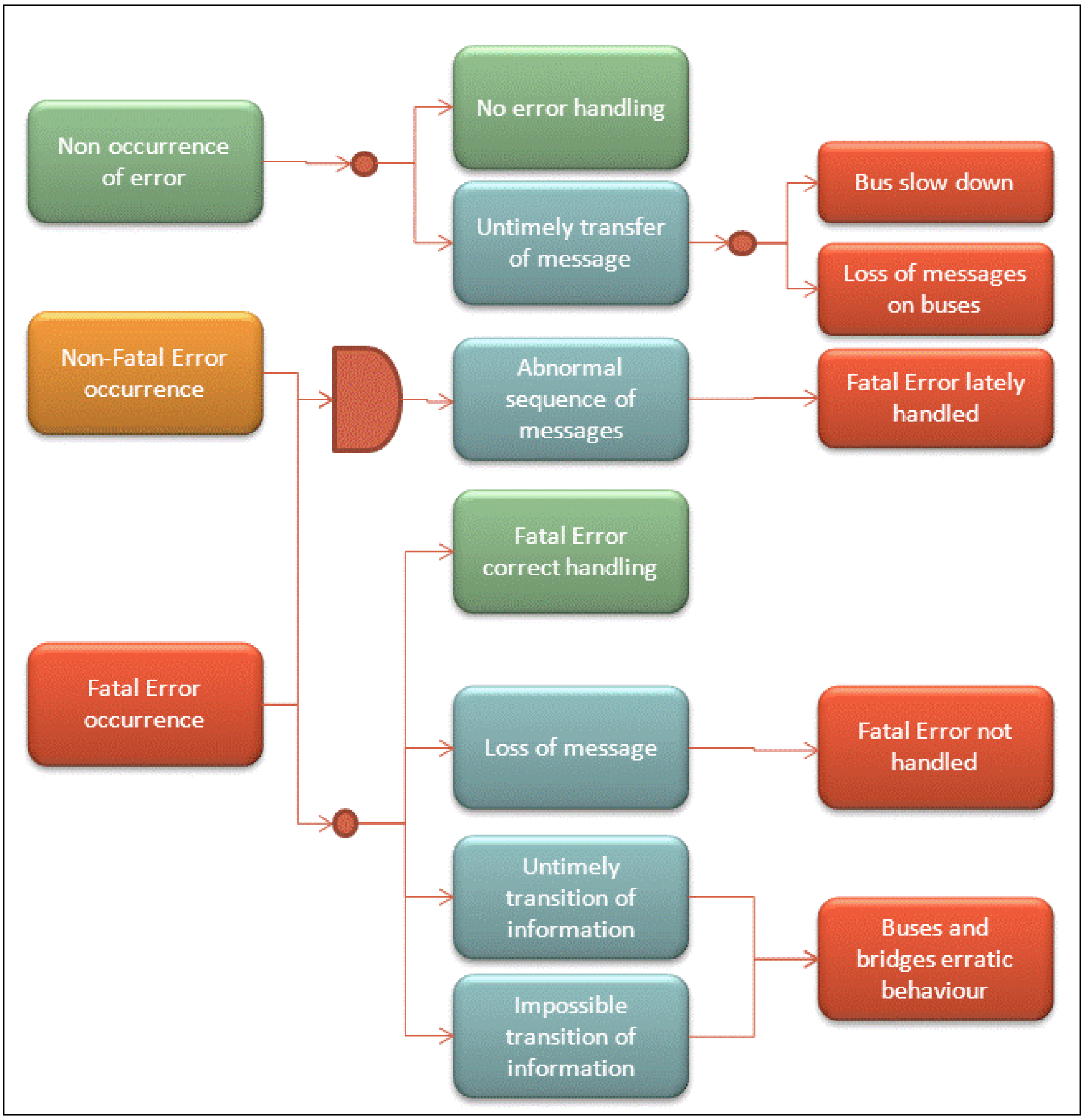

Non occurrence of error
No error handling
Untimely transfer of message
Bus slow down
Loss of messages on buses
Non-Fatal Error occurrence
Abnormal sequence of messages
Fatal Error lately handled
Fatal Error correct handling
Fatal Error occurrence
Loss of message
Fatal Error not handled
Untimely transition of information
Impossible transition of information
Buses and bridges erratic behaviour

Figure 65 : summary of error handling block possible failures;

On this diagram:

- The states on the left side summarize the errors (or non-error) that can occurs on the interface blocks (this errors correspond to severity level settled version of errors encountered on Figure 63 and Figure 74);

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- The central failures are the standard classification scheme used in this report – in green the absence of failure (normal functioning), in grey the standard failures. They can be considered as the state of the error handling block;
- The states on the rights represent the overall communication ensemble (as depicted on Figure 52);
- AND gate have to be understand as "in case of occurrence of A and B";
- Bullet shall be considered as non-exclusive branching condition "A and/or B could occur".

8.2.3.7.2.1 Loss of message

Loss of an error may appear due to some error of the block or of some configuration so that error does not result in any internal or external sanction. Such behaviour may cause transmission of erroneous frame without alert.
If one considers that transmission of erroneous frame is due to some errors or fault elsewhere, such a scenario implies two errors (on the frame transmission and on the error handling block). If it is considered that frame transmission randomly suffers from some errors[42] then error handling loss can directly impact safety stakes.

In chapter 9, we formalize the importance to test the equipment with monitoring of the integrated error mitigation mechanisms in order to know if covered failure may occurs in operation or not.

8.2.3.7.2.2 Untimely transfer of message

Untimely detection of error will cause loss of frame and performance slow down by occupation of the PCIe devices. It may also, by the fact, cause untimely retry and thus untimely transmission of messages.

8.2.3.7.2.3 Abnormal sequence of messages

Abnormal sequence can invert fatal and non-fatal error treatment so that fatal error treatment is abnormally delayed with consequences close to the loss already discussed. Such behaviour could be caused by wrong configuration for "error severity level setting" of **Figure 63**.

8.2.3.7.2.4 Untimely or forbidden transition of information

Even if error is raised in time it can suffer from untimely or forbidden transition of information. Such information can be the associated severity level, the origin of error (which bus), the nature of error on the bus … An untimely transition of such information will cause an erratic behaviour of the system depending upon the failure type. For instance if the error origin suffer an untimely transition, then the sanction will be addressed to the wrong bus so that messages will be lost on the incriminated interface and faulty messages can continue to be sent on the faulty interface. In case of forbidden transition, the error will be lost.

8.2.3.7.2.5 Impossible transition of information

Information is stuck so that error characterisation never changed may cause the same behaviours as those already described on untimely or forbidden transition of information.

[42] Faulty frame sending is then a particular life situation.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.3.7.3 Intrinsic mechanisms

No intrinsic mechanism seems to be implemented in order to mitigate errors of the error handling block.

8.2.3.8 Reset

8.2.3.8.1 Description

TSI 384 reset feature is driven by different entries summarized on Figure 66:

- A standardised reset message transmit on the PCIe;
- A standardized DL_Down state of the PCIe bus[43];
- A configuration message transmit on the PCIe and setting a particular bit in the Configuration Space Register (CSR) to "reset";
- A GPIO entry PCIe_PERSTn.

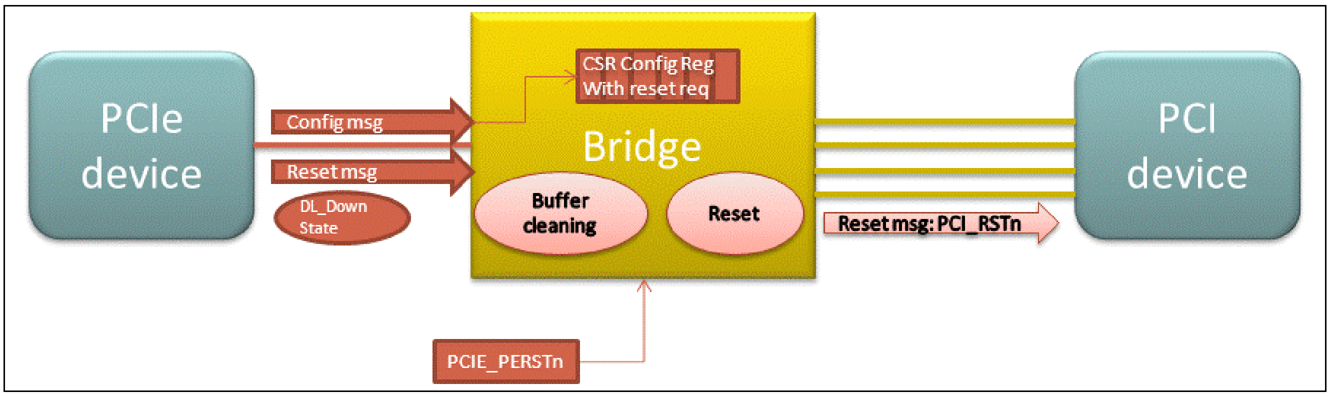


Figure 66 : Bridge level reset feature summary;

When activated, these four entries trig three levels of reset founded on standardized PCIe reset states [32], [36]:

- Level 0 warm (at power up and without power down phase) and cold reset (with a power cycle up-down-up);
- Level 1 hot reset;
- Level 2 PCI reset only;

These different types of resets act on:

- Bridge reset with internal register partial or total initialisation;
- PCI reset;
- Bridge traffic draining and TLP request dropping.

If we consider the Reset as a block it should have the interfaces presented on Figure 67.

[43] DL_down state (DL for Data Link) means that the bridge has lost communications at the physical or data link layer with the upstream device.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

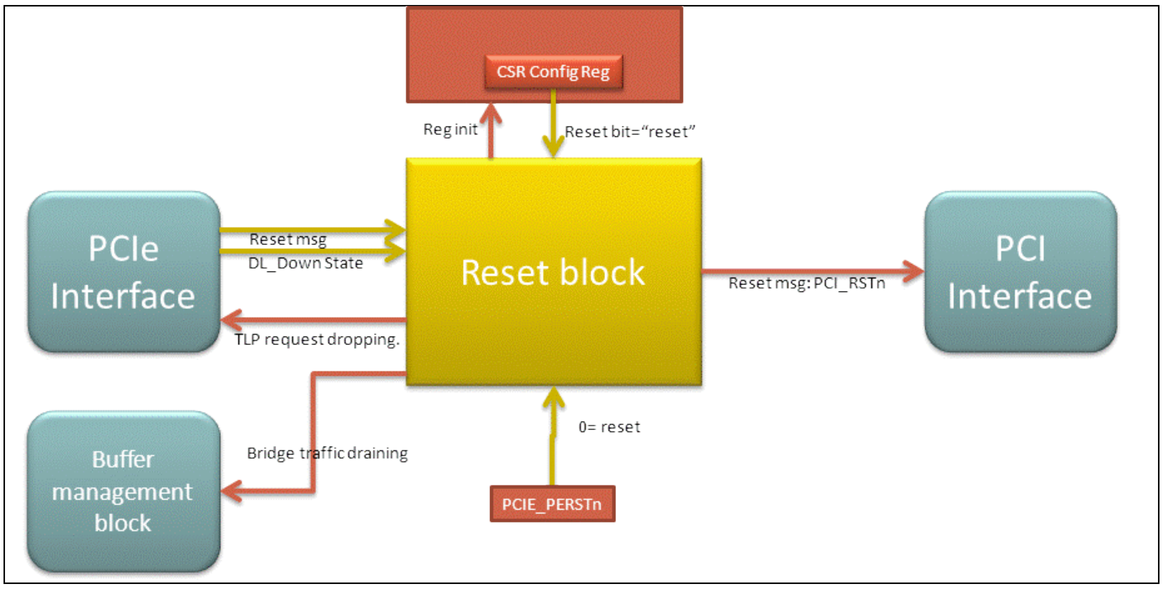


Figure 67 : Context diagram of Reset Block;

From the context diagram of **Figure 67** it is possible to construct a state diagram (**Figure 68**) with on the left side the states of the input interfaces, in the central part the state of the reset block and in the right side the state of the outputs.

On this diagram the sticky registers excluded from hot request concern typically many Advanced Error Reporting registers [36].

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

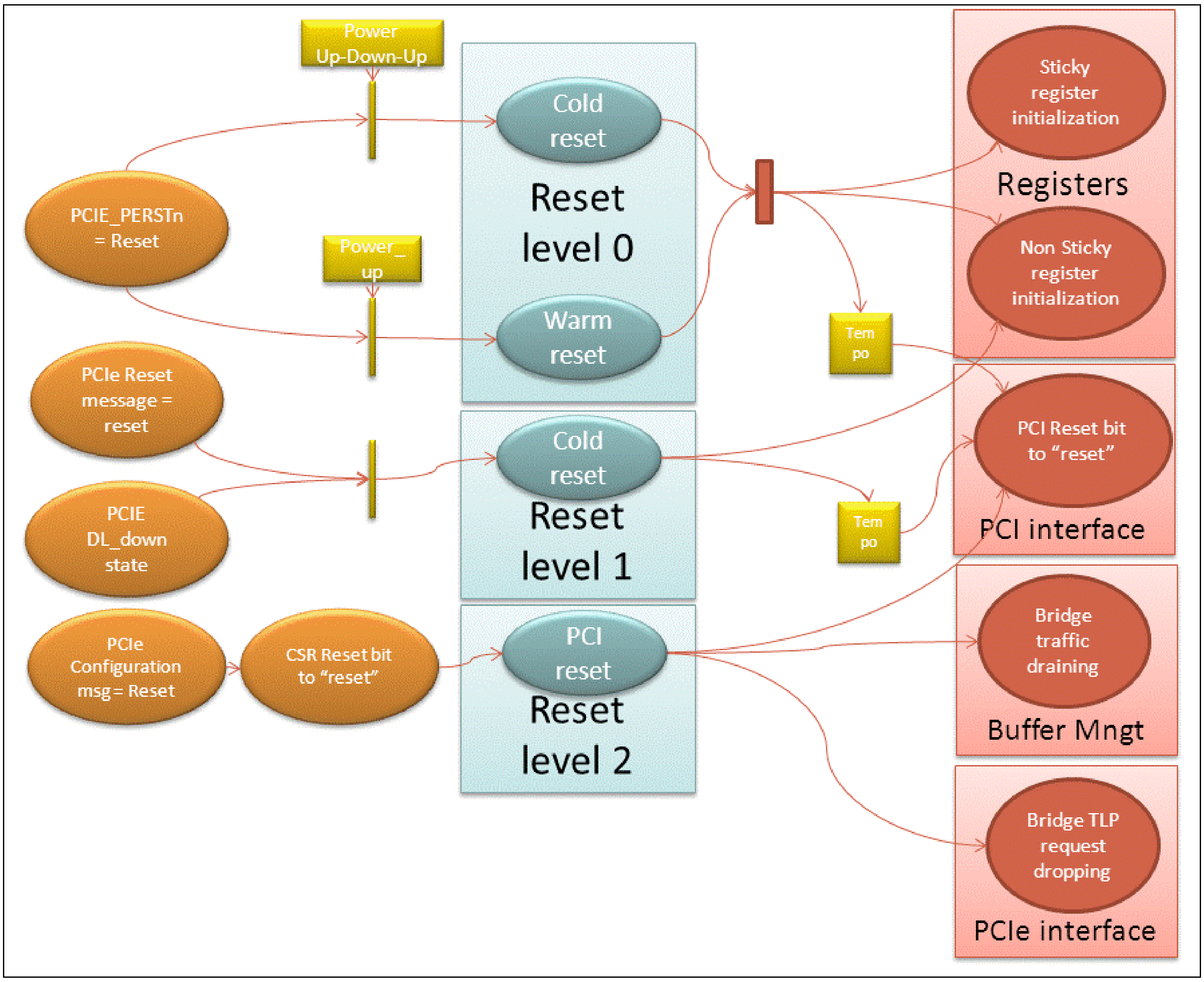


Figure 68 : Reset state diagram of the bridge;

Summary: Reset is basically triggered by external GPIO or by PCIe event or requests. It leads to different level internal reset of the bridge and PCI bus (level 0 and 1) and PCI bus alone (level 2) with some coherency action on PCIe interface (e.g. Transaction Layer cleaning)

8.2.3.8.2 Failure modes

On the basis of **Figure 67** and **Figure 68** the following failure analysis can be performed, considering the failures of the reset block output.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.3.8.2.1 Loss of message

A loss of all of the block output will result in an impossible activation of reset mode. This imply an impossibility to restart the bridge in case of major failure (level 0 or 1) or an impossibility to reset the PCI link (level 2).

In case of loss of one to three over four outputs, the reset coherency cannot be ensured and the PCI/PCIe integrity will not be guaranteed. For instance it could occur that:

- The device restarts with an old configuration. This configuration could be corrupted if it is the cause of the reset but could simply be no longer applicable;
- The TL buffers and the internal buffers and queues of the buffer management block could be full of transaction waiting acknowledgement from the PCI. These acknowledgments will never be received because the PCI reset;
- Etc.

Depending where the error occurs in the block, it can be global (for instance if the signal PCIE_PERSTn is permanently set to 1) or local to one output.

8.2.3.8.2.2 Untimely transfer of message

An untimely transfer of reset block output will lead to a partial (or total) untimely reset.

- In case of total untimely reset, the link will restart after a latency but the Maximum Execution Time could be impacted;
- In case of partial (one to three outputs over four), the effect is similar to those described in the case of partial loss (paragraph 8.2.3.8.2.1.);

8.2.3.8.2.3 Abnormal sequence of messages

An abnormal sequence of messages should not have significant effects as the bridge wait that all the operations are ended before to start correctly.

8.2.3.8.2.4 Untimely or forbidden transition of information

In case of untimely transition of one or more reset information some behaviour could occur that look like behaviour described in preceding paragraphs. In particular in case of untimely transition of PCI reset bit to “reset”, the PCI will not reset in coherent manner with others signals.

8.2.3.8.2.5 Impossible transition of information

Similar behaviours will occur in case of impossible transition. For instance, in case of impossible transition of PCI reset bit to “reset” the PCI reset will appear to be impossible even if all other signals are sent to reset.

8.2.3.8.3 Intrinsic mechanisms

No intrinsic mechanisms cover the preceding failure modes.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.2.3.9 Debug interface

8.2.3.9.1 Description

Debug interface allows deep diagnosis of the bridge in development, testing, integration and maintenance of the overall equipment. On a bridge, debug interface features cannot be used as particular diagnostic feature in operation. It is thus considered that debug interface has no interaction with safety related behaviours.

8.2.3.9.2 Failure modes

Not applicable

8.2.3.9.3 Intrinsic mechanisms

Not applicable

8.2.4 Concluding remarks

It is noticeable that a bridge can develop the complete panel of failure modes and does not seem to contain any internal detection / mitigation mechanisms. Indeed all the failures described on bridge blocks results in failures on PCI or PCIe messages listed in chapter 7. Bridge will rely on pure architectural mechanisms in order to detect and mitigates these failures (see section 9.3.4.1, 9.3.4.2, 9.3.4.3 and **Erreur ! Source du renvoi introuvable.**).

8.3 ARINC 429 INTERFACE DRIVERS

8.3.1 Introduction and available data

Several types of A429 interfaces driver are available with different level of complexity. Among them are of particular interest:

- The HOLT HI-3585/3586,
- The DDC DD-00429.

Each of these interface drivers is described by a datasheet. Use of A429 standard is also of interest.

Document name	Document description	Examples
Datasheet	Datasheet present the main features and characteristics of the DDR memory : • Functional description, • Functional block diagram and state diagram, • Electrical specification • Mechanical specification • Thermal characteristics • Etc.	HOLT HI-3585/3586 [37] DDC-00429 [38] and DDC-42900 [39]
Standard	ARINC 429 standard	[40]

Table 8: A429 driver list of reference documents

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

8.3.2 Architecture description

The basic function of ARINC 429 driver is to receive or transmit (or both) and to convert received information into typical format found on boards: SPI, PCI, local bus, etc.
In this sense the driver can be considered as a bridge. As ARINC 429 is very prescriptive (many information carried by the frame are prescript by the protocol) the driver may implement protocol verifications.

Due to the ARINC 429 bandwidth (bus frequency from 12.5kHz to 100kHz) and the few interfaced buses on COTS components (less than eight), either SPI bus or low end parallel local bus are selected for processing core connection to cope with a few megabit per second bandwidth.

The issue for the COTS manufacturer is more a small component package issue and the need to mix in a same device analogue (differential +/-10V with common mode and lightning strike residual high voltage expectations) and digital (3.3V to 5V range) parts.

HOLT HI-3585/3586 Datasheet [37] proposes the following block diagram (Figure 69).

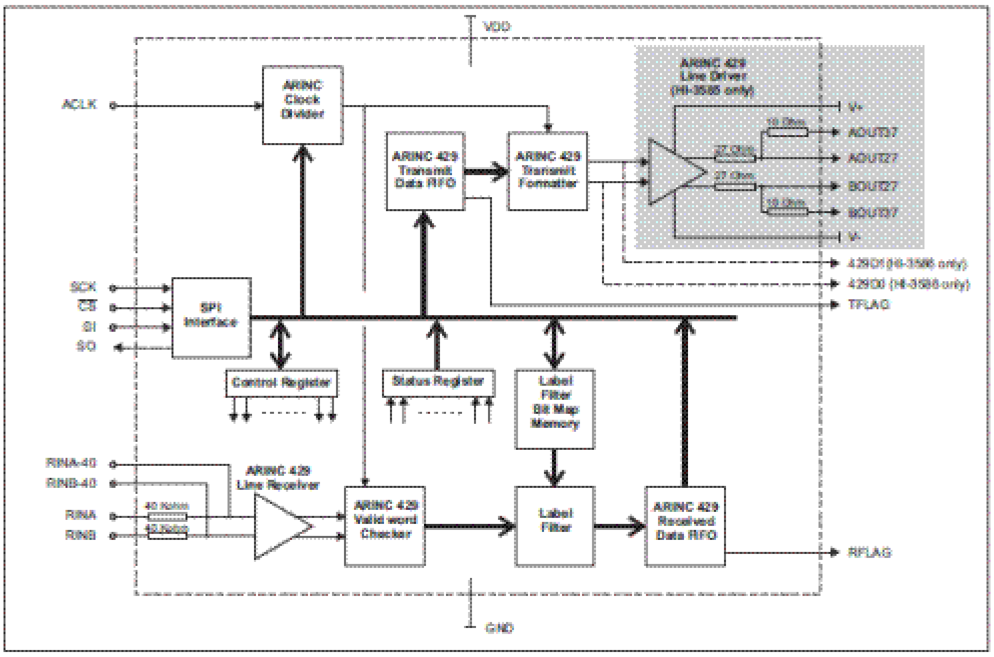


Figure 69: HOLT HI-3585/3586 block diagram;

The interface of the HI-3585 driver with the external world is performed by a SPI bus (on the left side of Figure 69). The transmission levels are on the top part of the diagram and the receiving level on the bottom part. HI-3585 ensures minimum check features on the words.

DD-00429 datasheet [38] proposes the block diagram depicted on Figure 70.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

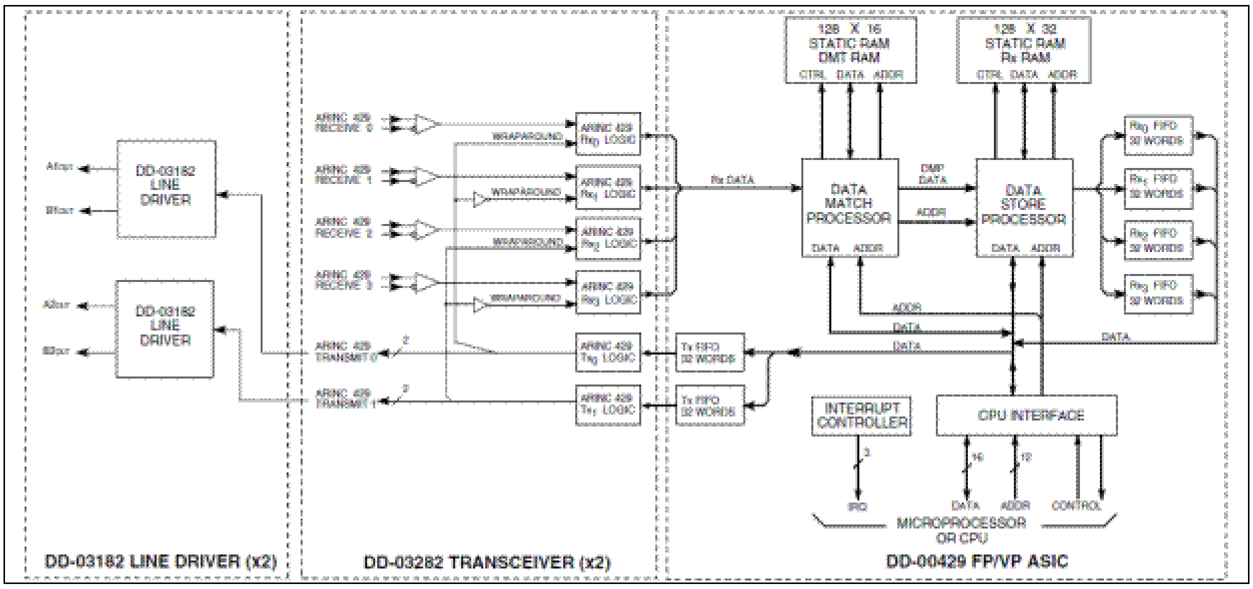


Figure 70 : DD-00429 ASIC with external analogue adaptors block diagram;

This A429 driver appear to be more complex than HI-3585. It communicates with the CPU on a local bus (bottom-right of Figure 70) and has several lines in emission and reception (left part of the diagram) . The conformity of received frames is realized by a controller as well as the storage of received frames .

In order to study a typical A429 driver a generic model is proposed on Figure 71.

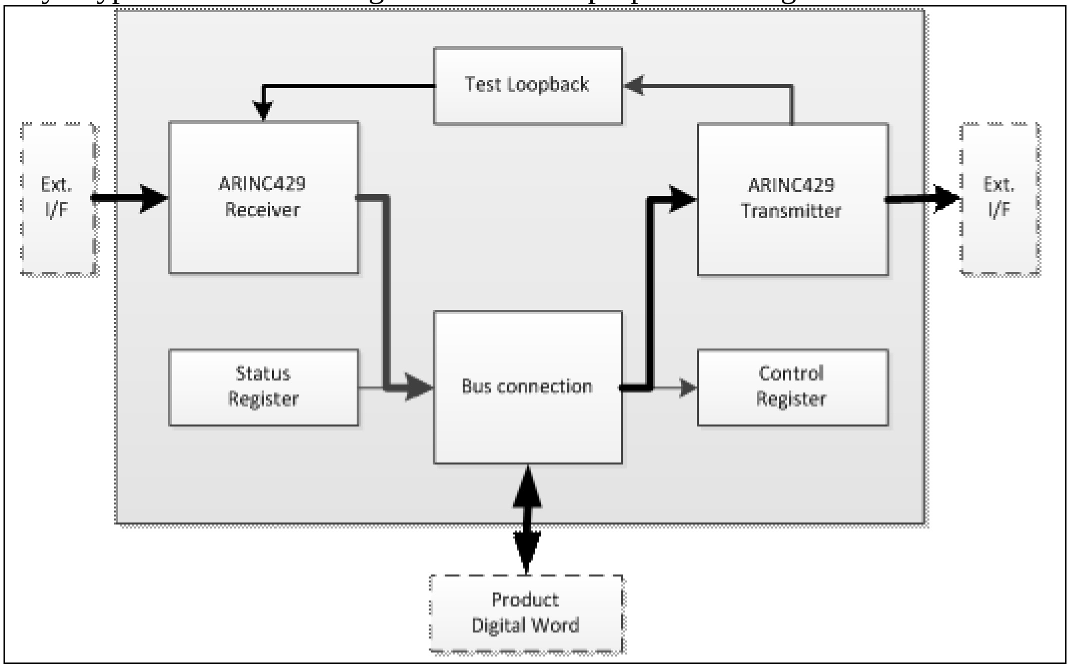


Figure 71 : Generic block diagram of A429 driver;

List of ARINC 429 COTS provided features:

- By Control Register:
    - Device configuration (external input clock, ARINC bit-mapping on SPI bus, master reset, test loopback mode…);

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Receive lines configuration (speed, label to store, parity check);
- Transmit lines configuration (speed, parity generator);

- By ARINC 429 receiver:
  - Optional Analog to Digital receive adaptation;
  - Receive checks (gap size, bit rate, parity, label to store);
  - Label filtering
  - Label storage address computing (unless a by default FIFO protocol is implemented that let software to cope with this function)
  - Receive data storage;
- By ARINC 429 transmitter:
  - Transmit data storage;
  - Scheduling of the transmission (bit rate, parity bit, …);
  - Optional Digital to Analogue transmit adaptation ;
- By Bus connection:
  - Management of the SPI slave bus interface or of the parallel local bus interface to host processor while outing interruption signal;
  - Access to the control register:
    - Device configuration (external input clock, ARINC bit-mapping on SPI bus, master reset, test loopback mode…);
    - Receive lines configuration (speed, label to store, parity check);
    - Transmit lines configuration (speed, parity generator;
  - Access to the status register;
- By Test Loopback:
  - The transmitter's digital outputs are internally connected to the receiver digital inputs.

8.3.3 Failure modes

The following main streams have to be considered when discussing about ARINC 429 communication failures:

- Receive data stream: Stream from the ARINC 429 receiver bus to the COTS local bus connection

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Transmit data stream: Stream from the COTS local bus connection to the ARINC 429 transmitter bus

The streams from/to the COTS local bus to the control register / from the status register are enablers for the ARINC 429 communication and can also affect the two main communication streams.
A last stream exists in the component: COTS internal test loopback connection can be established between the digital parts of ARINC 429 transmitter and emitter.
Most of the ARINC 429 COTS propose this embedded Test Loopback from the digital transmit part to the digital receive part. While enabling to check most of digital buried blocks during a scheduled test phase, this mechanism, if failed or active during operational mode, can induce combined failures on both transmit and receive blocks, independently of any local bus access.

Each of the failure modes of the subchapter 6.3 are declined on messages that come from (outgoing message) or that go to (incoming message) more buried blocks compared to ARINC 429 COTS.

8.3.3.1 Loss of message

Receive data stream
In case of failure of the ARINC 429 analogue to digital conversion, a loss of message will be more likely experienced at received data memory level as it seems not foreseeable to have voltages erroneous detection while maintaining respect to the bit rate timings due to the intrinsic failure mitigation incorporated in the ARINC 429 physical and logical layers.

In case of COTS external clock loss, loss of message will be experienced as the COTS is no more able to sample the incoming message.

Transmit data stream
In case of failure of the ARINC 429 digital to analogue conversion, a loss of message will be more likely experienced as it seems not foreseeable to have erroneous voltage levels while maintaining respect to the bit rate timings.

In case of COTS external clock loss, loss of message will be experienced as the COTS is no more able to schedule the outgoing message.

8.3.3.2 Untimely transfer of messages

Receive data stream
In case of an erroneous configuration (e.g. with not intended labels to catch), a receiver FIFO like memory can be saturated with data of unintended labels. This can lead to modify the received sequence of messages with intended labels to catch, even with message loss.

Transmit data stream
In case of an erroneous management of the transmit FIFO memory, a collection of messages can be emitted continuously without refresh of the data (i.e. babbling).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.3.3.3 Abnormal sequence of message

Receive data stream
This failure mode can occurs when data are stored in some buffer or memory before read through the local bus. It is in particular the case when the FIFO management counters are corrupted in ARINC 429 queuing receive mode (where a same label is used for a data collection, it's the receive order of the data that give sense to the data collection extracted from the messages).

Transmit data stream
This failure mode can occurs when data are stored in some buffer or memory before write on the transmit line. It is in particular the case when the FIFO management counters are corrupted in ARINC 429 queuing transmit mode (where a same label is used for a data collection, it's the transmit order of the data that give sense to the data collection extracted from the messages). A collection of messages can be emitted with a not consistent age between messages or with missing messages in the collection.

8.3.3.4 Untimely or forbidden transition of information

Receive data stream
This topic corresponds to the transition to a wrong value of data or label values.
In case of COTS with multiple receive lanes; a failure can occur that wraps the labels to catch from one lane to the others. If the COTS uses the label and lane rank to store the data in an RAM, this case will can be particularized in "impersonation".
A simpler case is a corruption of a receive memory cell.

Transmit data stream
This topic corresponds to the transition to a wrong value of data or label values.
In case of COTS with multiple transmit lanes; a failure can occur that wraps the messages to transmit from one lane to the others. If the COTS uses the label and lane rank to store the data in an RAM, this case will can be particularized in "impersonation".
A simpler case is a corruption of a transmit memory cell.

8.3.3.5 Impossible transition of information

Receive data stream
In case of loss of addressing capability on the local bus, all read accesses will return the same dummy data on the bus.

Transmit data stream
In case of loss of addressing capability on the local bus to the transmit memory, all write accesses will be non-effective and a dummy data stuck in the memory.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.3.4 Failure Detection & Mitigation

Failure detections and mitigations are those of the ARINC 429 physical and data-link layers.

8.3.5 Concluding remarks

Errors in ARINC 429 interface driver design results either in failures on the ARINC 429 messages (see section 7.4) and/or failures on local communication bus (for instance SPI) with the other architecture building block (in general a microprocessor). These errors are partly covered by the ARINC 429 detection and mitigation means and by architectural mechanisms such as output monitoring (subsection 9.3.4.1.3).

8.4 MIL-STD-1553 INTERFACE DRIVERS

8.4.1 Introduction and available data

MIL-STD-1553 interfaces drivers are described in a Datasheet that describes the architecture, features, electrical connexion and environmental constraints data. Table 9 lists the reference documents of interest in order to perform the study of the component.

Document name	Document description	Examples
Product Brief	Product brief introduce to the features covered by the COTS.	PCI-Express AceXtreme® Product Brief [21]
Datasheet	Datasheet present the main features and characteristics of the DDR memory : • Functional description, • Functional block diagram and state diagram, • Electrical specification • Mechanical specification • Thermal characteristics • Etc.	PCI-Express AceXtreme® Datasheet [41]
Standard	Standard	MIL-STD-1553B notice 4, [19]

Table 9: MIL-STD-1553 drivers, list of reference documents

8.4.2 Architecture description

Figure 72 presents the block diagram of DDC AceXtrem MIL-STD-1553 driver.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

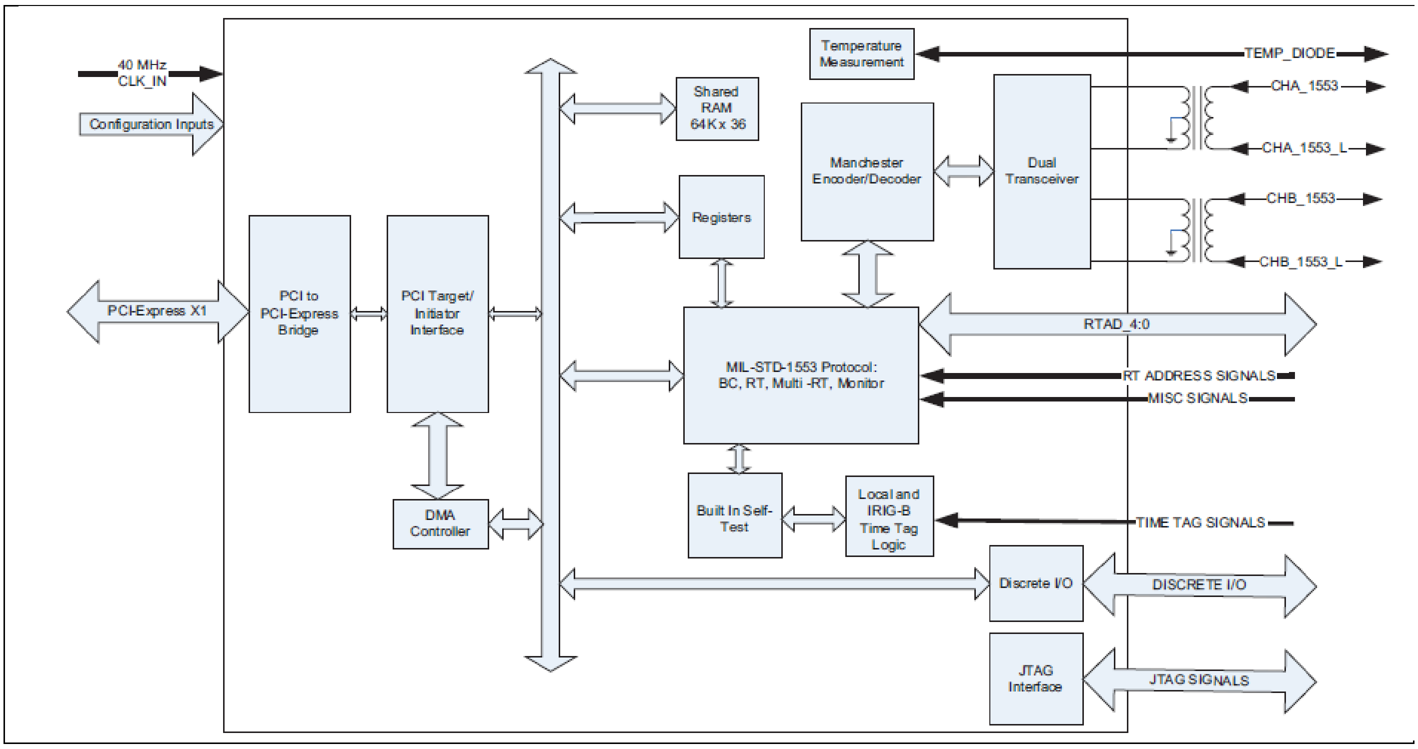


Figure 72: DDC AceXtreme block diagram from [41].

This block diagram shows the two redundant lines on the top right coupled with dual transceiver and Manchester encoder/decoder, to a protocol monitor and to a local bus. The PCIe I/O is realized through a bridge PCI-PCIe. An equivalent model is presented on Figure 73 (a) (transceivers have been decoupled).

Even if this COTS covers a larger perimeter than the connection of MIL-STD-1553 interface to a parallel local bus, extra blocks is excluded from the studies of following paragraphs.

The resulting studied model for a MIL-STD-1553 driver is given on Figure 73 (b).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

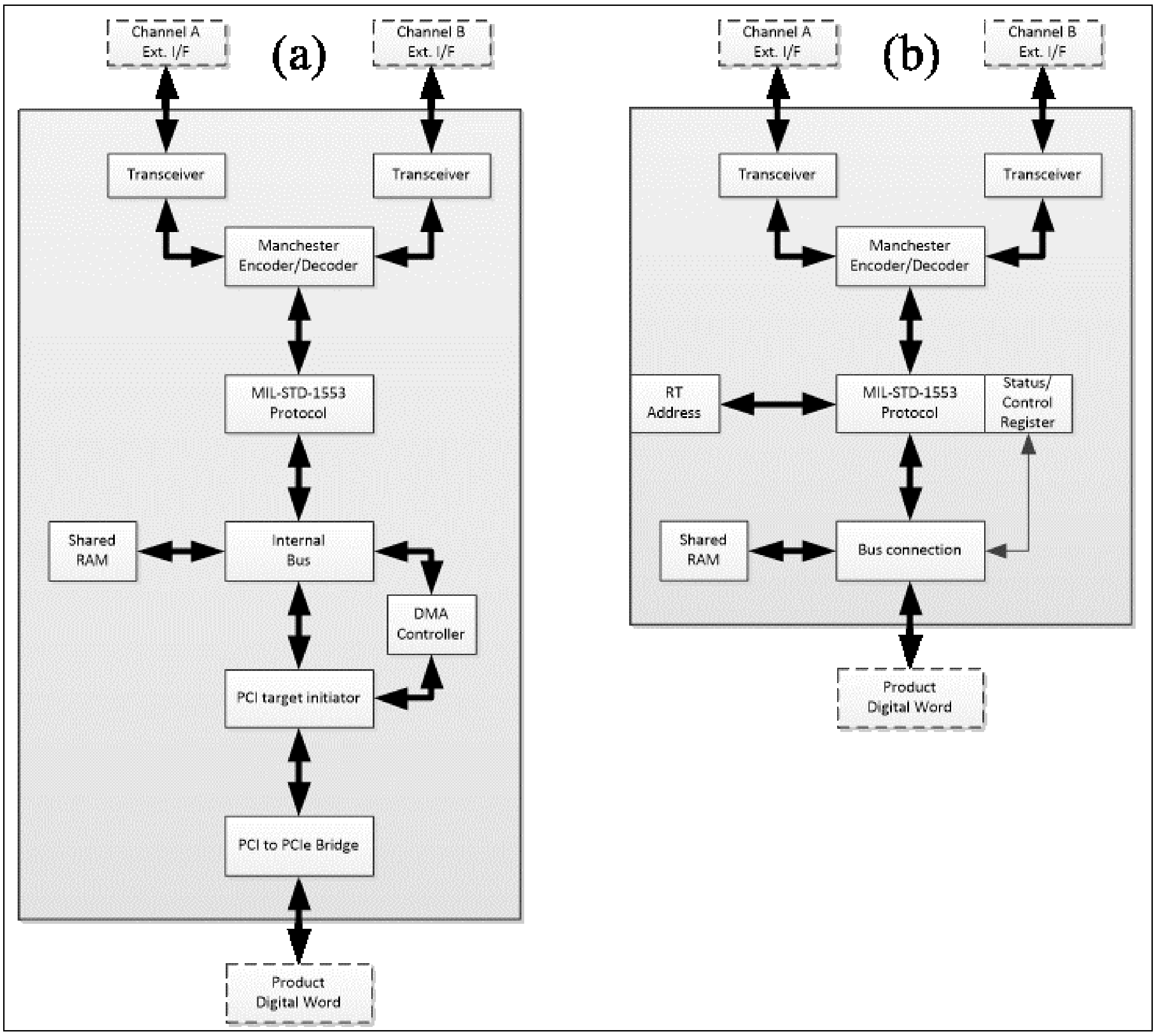


Figure 73: MIL-STD-1553 Driver Block diagram

List of MIL-STD-1553 COTS functions of interest:

- Device configuration (MIL-STD-1553 mode selection, …)
- Analogue to digital and digital to analogue adaptations
- Manchester encoding/decoding layers
- 1553 protocol layers, in line with mode selection and Remote Terminal address
- Transmit lines configuration (speed, parity generator)
- Management of the internal Random Access Memory
- Management of the local bus connection

8.4.3 Failure modes

The following main streams have to be considered when discussing about MIL STD 1553 communication failures:

- Channel A receive/transmit data stream: Stream from the Channel A bus to the shared RAM

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

- Channel B receive/transmit data stream: Stream from the Channel B bus to the shared RAM
- COTS local bus from/to shared RAM data stream: all the MIL-STD-1553 incoming/outgoing communication stream of both A and B channels flows through the shared RAM to be read/write by a processing core while accessing to the COTS local bus data flow

The streams from/to the COTS local bus to the control register / from the status register are enablers for the MIL-STD-1553 communication and can also affect the three main communication streams.
A last stream exists in the component: a COTS external port allows access to 5 hardwired programming pins with one parity protection bit and so to COTS Remote Terminal address on the MIL-STD-1553 bus.
Most of the failures modes seen on ARINC 429 COTS can be experienced on MIL-STD-1553 COTS and the analysis focuses on MIL-STD-1553 specific failures modes.

Each of the failure modes of the subchapter 6.3 is declined on messages that come from (outgoing message) or that go to (incoming message) more buried blocks compared to MIL-STD-1553 COTS and that ensure responsibilities in higher layers.

As for all memory structures, most of the following failures can be experienced.

8.4.3.1 Loss of message

Channel X receive/transmit data stream
Most likely failure of the MIL-STD-1553 analogue to digital or digital to analogue layers will lead to a loss of message due to the intrinsic robustness of the MIL-STD-1553 logical and physical layers.

COTS local bus from/to shared RAM data stream
Erroneous management of receive buffers that are erroneously overwritten.

8.4.3.2 Untimely transfer of messages

Channel X receive/transmit data stream
In case of corruption of its local RT address, the RT can respond to a BC command addressed to the corrupted RT address and untimely transfer of messages (i.e. impersonation). If no other RT shares the corrupted RT address, the MIL-STD-1553 communication will be effective. If another RT shares the corrupted RT address, a collision will most likely happen between the two RTs. The intrinsic robustness of the communication layer will lead to response interruption after collision detection.

COTS local bus from/to shared RAM data stream
Erroneous management of receive buffers that are erroneously partially corrupted.

8.4.3.3 Abnormal sequence of message

Channel X receive/transmit data stream

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

This failure mode can occur when data are stored in some buffers of the shared RAM and the management of the buffers experienced failures. Data buffers can be routed either to a bad channel, or as a response to a bad command.

COTS local bus from/to shared RAM data stream
Erroneous management of a receive buffers in case of messages queuing that does not respect the messages ordering.

8.4.3.4 Untimely or forbidden transition of information

Channel X receive/transmit data stream:
The simple corruption of a shared RAM cell can lead to such a failure mode.

COTS local bus from/to shared RAM data stream
The simple corruption of a shared RAM cell can lead to such a failure mode.

8.4.3.5 Impossible transition of information

Channel X receive/transmit data stream
In case of inadvertent switching from Bus Controller mode to either Remote Terminal or Bus Monitor modes, as the unique BC of the bus disappears, transition of information on the bus becomes impossible. Both channels will experience this failure in case of configuration failure and so the channels redundancy will become ineffective.

COTS local bus from/to shared RAM data stream
In case of loss of addressing capability on the local bus to the shared memory, all read or write accesses will be non-effective and a dummy data stuck in the memory.

8.4.4 Intrinsic robustness of the physical layer and failure mitigation mechanisms

Intrinsic robustness of the physical layer and failure mitigation mechanisms are those of MIL-STD-1553 physical, data link and network layers (sections 7.5.3 and 7.5.4)

8.4.5 Concluding remarks

Errors in MIL-STD-1553 interface driver design results either in failures on the MIL-STD-1553 messages (see section 7.5) and/or failures on local communication bus (for instance PCIe) with the other architecture building block (in general a microprocessor). These errors are partly covered by the MIL-STD-1553 detection and mitigation means and by architectural mechanisms such as output monitoring (subsection 9.3.4.1.4).

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

8.5 DDRX SDRAM MEMORIES

8.5.1 Introduction and available data

Memory chip and DDRx SDRAM in particular are principally described by a datasheet that covers the architecture, features, electrical connexion and environmental constraints data. It appears also important to use the JEDEC related standard that is considered as customer background knowledge by the manufacturer.

In the following paragraph we consider a typical DDR3 memory from Micron manufacturer.

Document name	Document description	Examples
Datasheet	Datasheet present the main features and characteristics of the DDR memory : • Functional description, • Functional block diagram and state diagram, • Electrical specification • Mechanical specification • Thermal characteristics • Etc.	Micron MT41J type DDR3 SDRAM datasheet [42]
Standard	JEDEC Standard associated with the memory used can be obtained freely under licence agreement on JEDEC solid state technology association[44].	DDR3 SDRAM Standard JESD79-3F [43]

Table 10: list of input in order to analyse a DDR memory;

8.5.2 Architecture description

JEDEC DDR3 standard [43] provides a state diagram that described the complete behaviour of the chip. This state diagram is reproduced in [42] associated with few block diagram (for different DDR3 references)[45]. These block diagram have all the same structure reported in the background of Figure 74.

[44] The JEDEC web site is accessible under http://www.jedec.org/

[45] Note that older versions of JEDEC DDRx standards contained a very similar functional block diagram “intended to facilitate user understanding”.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

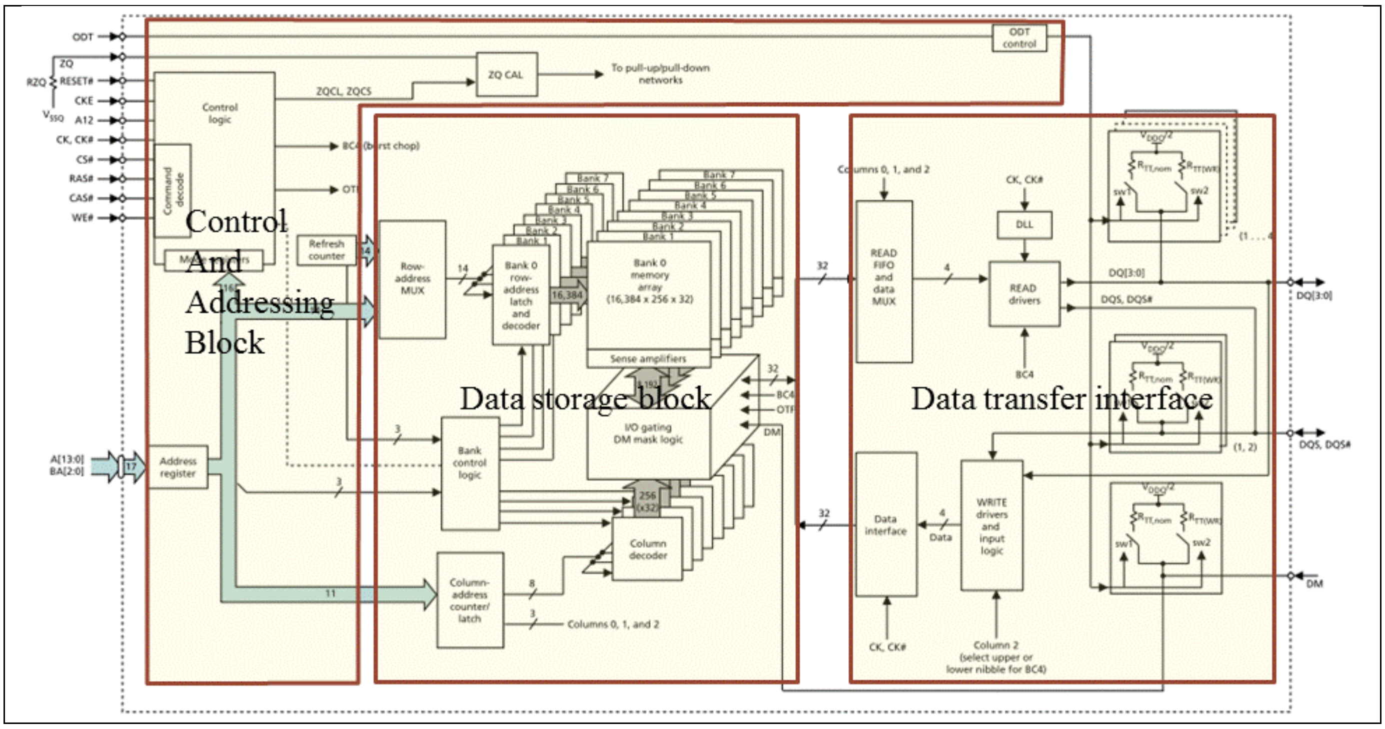


Figure 74: Internal architecture of a DDRx SDRAM (from [42]).

An abstraction level between the one proposed on Figure 9 of chapter 5 “state of the art” and the previous one is sufficient for our analysis. It proposes to split the DDR in three blocks as depicted on the foreground of Figure 74 and presented on Figure 75 here below.

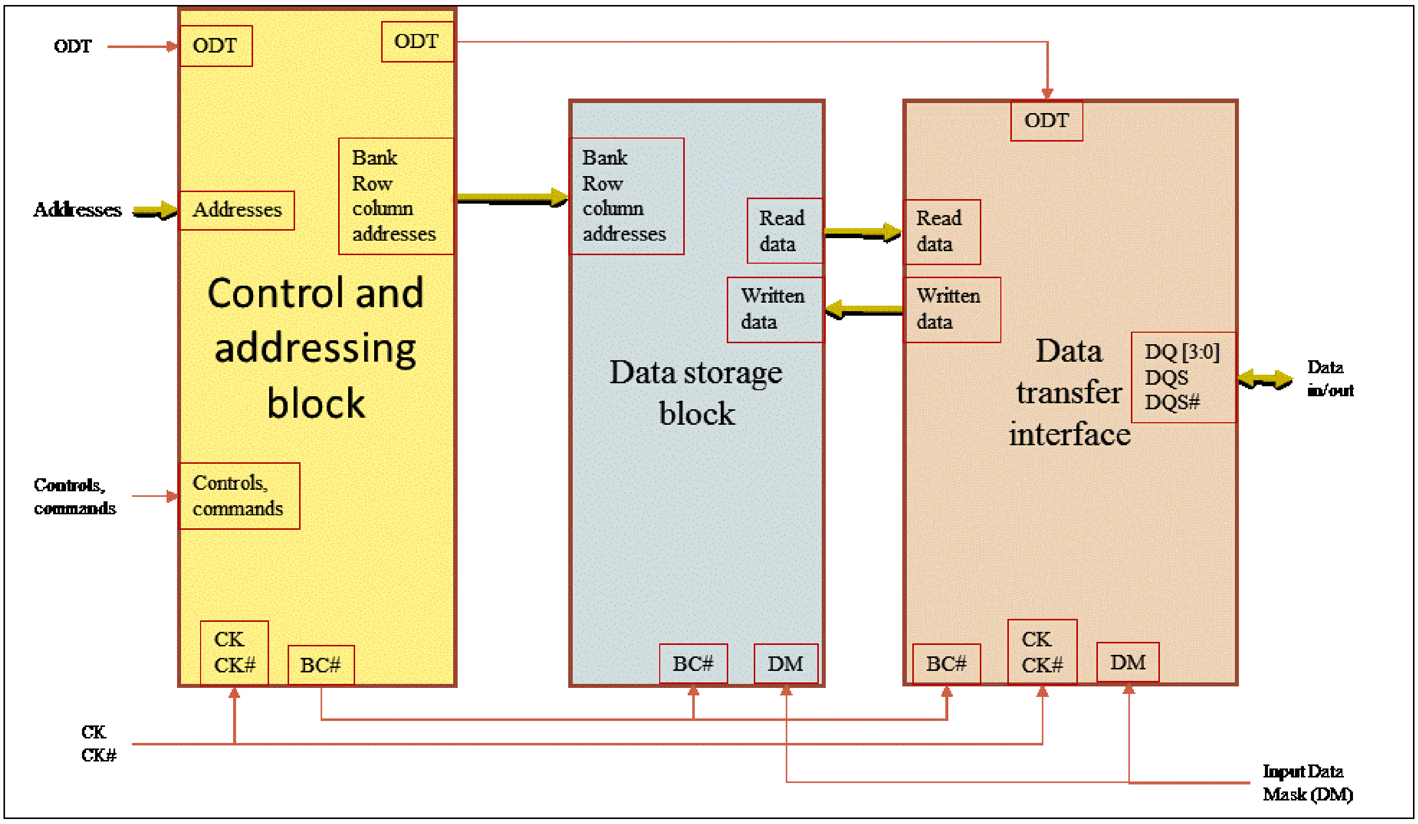


Figure 75: Simplified architecture of DDR SDRAM

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

The main messages used are summarize on Table 11

Flow name on Figure 75	Block	Corresponding flows on Figure 74	Description from [42] and [43]
ODT	Board OR Control and addressing block	ODT	On Die Termination: ODT enables termination resistance internal to the DDR3 SDRAM.
Addresses	DDR Controller (in context)	A[13:0] BA[2:0]	
Controls, Commands	DDR Controller (in context)	WE#, CAS#, RAS#, CS#, Reset#, CKE, ZQ, A12	WE#, CAS#, RAS# with the Chip Select CS# are command information for the memory. Reset# command the memory reset CKE is a Clock enable signal. It command the self-refresh operation of the DDR ZQ command the calibration mode
CK, CK#	DDR Controller (in context)	CK,CK#	
BC#	Control and addressing block	BC4	BC# (burst chop) transmit a burst command that reduce the burst elementary packet.
Bank, Row, Column addresses	Control and addressing block	Row address Column address counter Bank Control	Transfer to the data storage block the addresses of the data to be read or the address were data have to be written. Defines to which bank an Active, Read, Write, or Precharge command is being applied.
Read Data	Data Storage Block	Internal DDR flow not named	Transfer of read data from Data storage block to Data transfer interface
Written Data	Data Transfer Interface	Internal DDR flow not named	Transfer of written data from Data transfer interface to Data storage block
Data In/out	Data Transfer Interface OR DDR Controller (in context)	DQ DQS, DQS#	DQ: Data Input/ Output: Bi-directional data bus. DQS, DQS# (Data Strobe): output with read data, input with write data. Edge-aligned with read data, centred in write data. The data strobes DQS are paired with differential signals DQS# to provide differential pair signalling to the system during reads and writes. DDR3 SDRAM supports differential data strobe only and does not support single-ended.
Input Data Mask	DDR Controller (in context)	DM	During a Write access, input data is masked when DM is active coincident with that input data.

Table 11: DDR3 Blocks I/O;

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

In order to precise the failure modes and the corresponding behaviour of the blocks of Figure 75, the global behaviour of the memory should be described. In [42] and [43] a state diagram is provided.

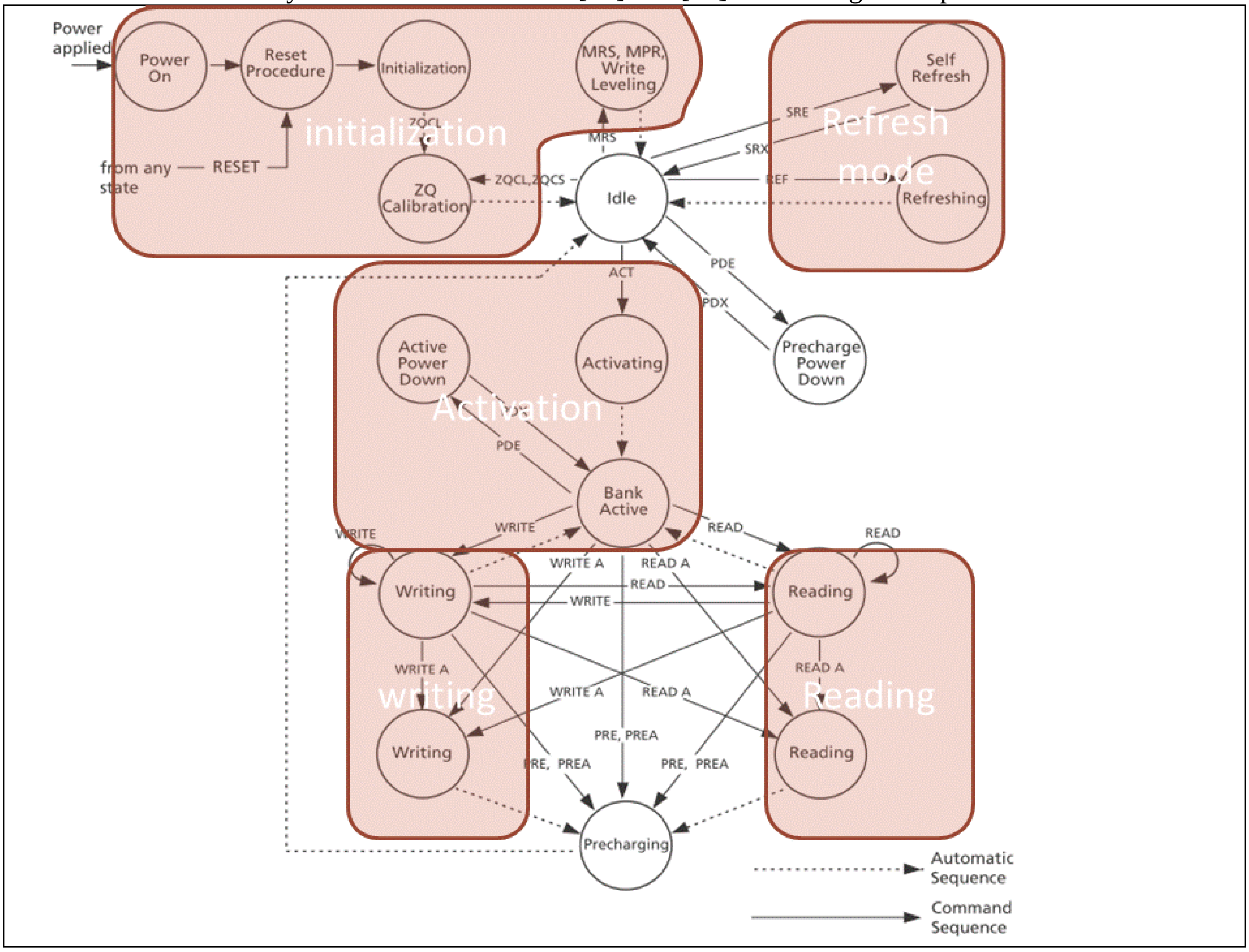


Figure 76 : DDR3 state diagram (from [42] and [43]). Principal zones have been highlighted.

On state diagram of Figure 76 the following groups of states appear:

- Initialization : states that are crossed during power on of reset sequence;
- Refresh mode: data refreshing process systematic for a SDRAM (see section 5.3.2.2); this refreshing process can be forced by the controller or periodically realized by the memory itself. Period decrease at temperature higher that 85°C;
- Activation: preparation of a reading or of a writing by pre-charging data in some buffer. Activation is triggered by memory controller order;
- Reading and Writing operations where memory push data to the controller or where controller push data to memory;
- Precharging: in order to be ready for next command memory.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

8.5.3 Block study

8.5.3.1 Control and Addressing Block

8.5.3.1.1 Description

This block mainly received the registration command from DDR controller that allows the activation of the memory. This registration is followed by commands that specify if the operation will be reading or writing.

When a reading operation is accepted by the memory this block received the address (Bank Address: BA and Row Address in the Bank: A) of the first row to be transferred. Transferred is realised by burst starting from this address while some controls stop it.

In order to ensure these functions, this block contains Mode Registers that configure the command and control applied.

8.5.3.1.2 Failure modes

8.5.3.1.2.1 Loss of message

The triggering of memory activation realized by the block is performed by discrete I/O so that it cannot be lost but misinterpreted. It is covered in paragraphs "*untimely or forbidden transition of information*" and "Impossible transition of information".

On the other hand, the banks, rows and columns addresses settled in output of this bloc to select the starting point of the burst can, due to some error, encounter a transition to a value that do not correspond to a valid address. In this case the address selection could be considered as lost. It will result at memory level an impossible realisation of read or write operation so that address transferred can be considered as lost (lower branch of Figure 77).

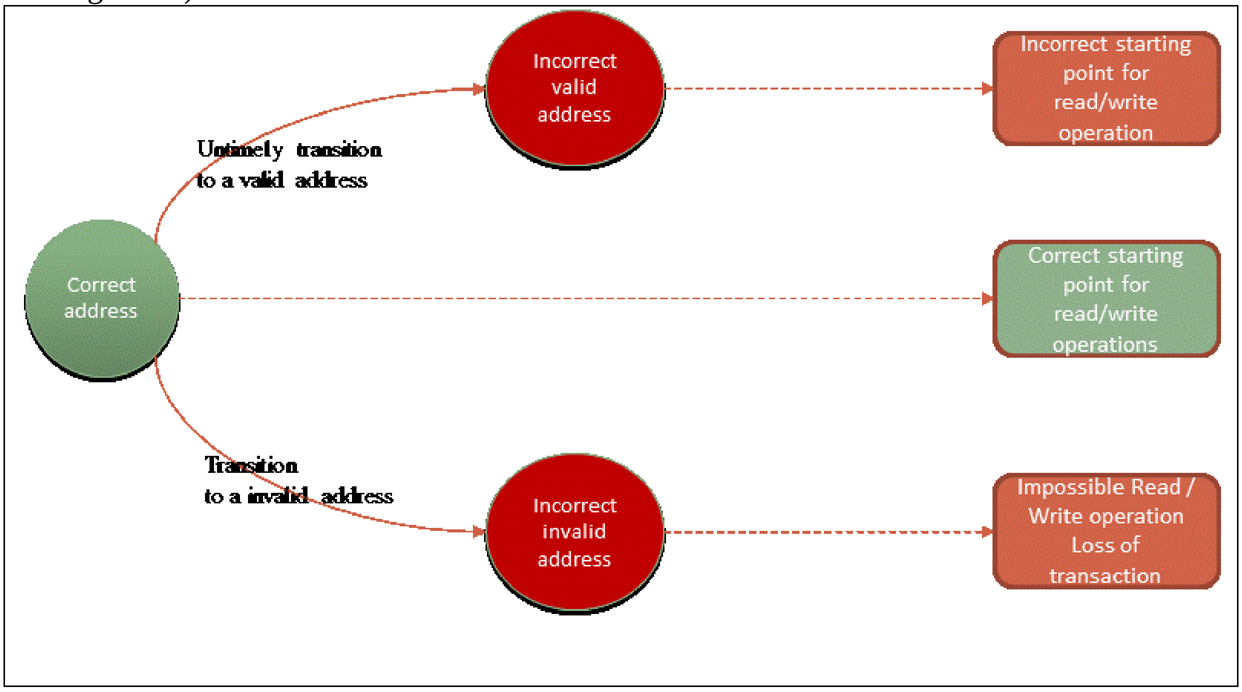

Incorrect valid address
Incorrect starting point for read/write operation
Untimely transition to a valid address
Correct address
Correct starting point for read/write operations
Transition to a invalid address
Incorrect invalid address
Impossible Read / Write operation Loss of transaction

Figure 77: Some possible failure of addressing operation. The incorrect valid address case is treated in forthcoming paragraphs.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.5.3.1.2.2 Untimely transfer of message

As in the previous paragraph, the binary nature of most of the signals make this failure mode non applicable.
In the case of addresses, it is possible to consider that this block untimely transfer address to the Data Storage Block so that the complete memory can never be considered in idle state. It could result of such behaviour that refresh cannot be trigger and that data could be lost.

8.5.3.1.2.3 Abnormal sequence of messages

Addresses are sent in parallel, each activating a bank, a column or a row. No ordering in address sequences seems to have an impact except if the bank address signal goes to 0 before acknowledgment of column and row address. This case could be more considered as an untimely transition of bank address to 0.

8.5.3.1.2.4 Untimely or forbidden transition of information

Due to the function it ensures and to its configuration, this block could make some command, controls or addresses transiting to wrong value. Such untimely transition could have different impact on other blocks and on the global memory behaviour. In particular, if command and control that govern the transition to active state suffer an untimely transition to active, it is possible that idle state could not be reach for a long time and that some refresh operation could not occur.

An untimely transition of address to a valid but incorrect address will make the reading or writing burst starting at the wrong address (upper branch of Figure 77). It may result of such a behaviour untimely transition of data sent to the processor to incorrect value and then wrong computation.

A forbidden transition of address to an incorrect address will cause the impossibility to write or read data.

The choice realised on Figure 74 and Figure 75 attached the On Die Termination (ODT) management to this block. It appears that, due to the low voltage level involved in DDR3 transfer, an error that could influence the ODT impedance so that ODT signal untimely changed could cause loss of data.

8.5.3.1.2.5 Impossible transition of information

Errors described in the case of "*untimely or forbidden transition of information*" could cause also impossible transition failure such as:

- Command or control that activates the memory for next transfer could suffer an impossible transition to activation. This could cause loss of data in write or read operation;
- Control that stops the burst transfer could also suffer an impossible transition to stop value so that burst could continue. Generating an abnormal occupation of the memory controller and possible loss of other data.

8.5.3.1.3 Failure mitigations

See paragraph 8.5.4.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

8.5.3.2 Data Storage Block

8.5.3.2.1 Description

Data storage block is made up of several banks of memories organized in column and rows. In order to access to right column and row in some bank, the block contains some logic that control the access to these elements on the basis of addresses received from Control and Addressing Block.

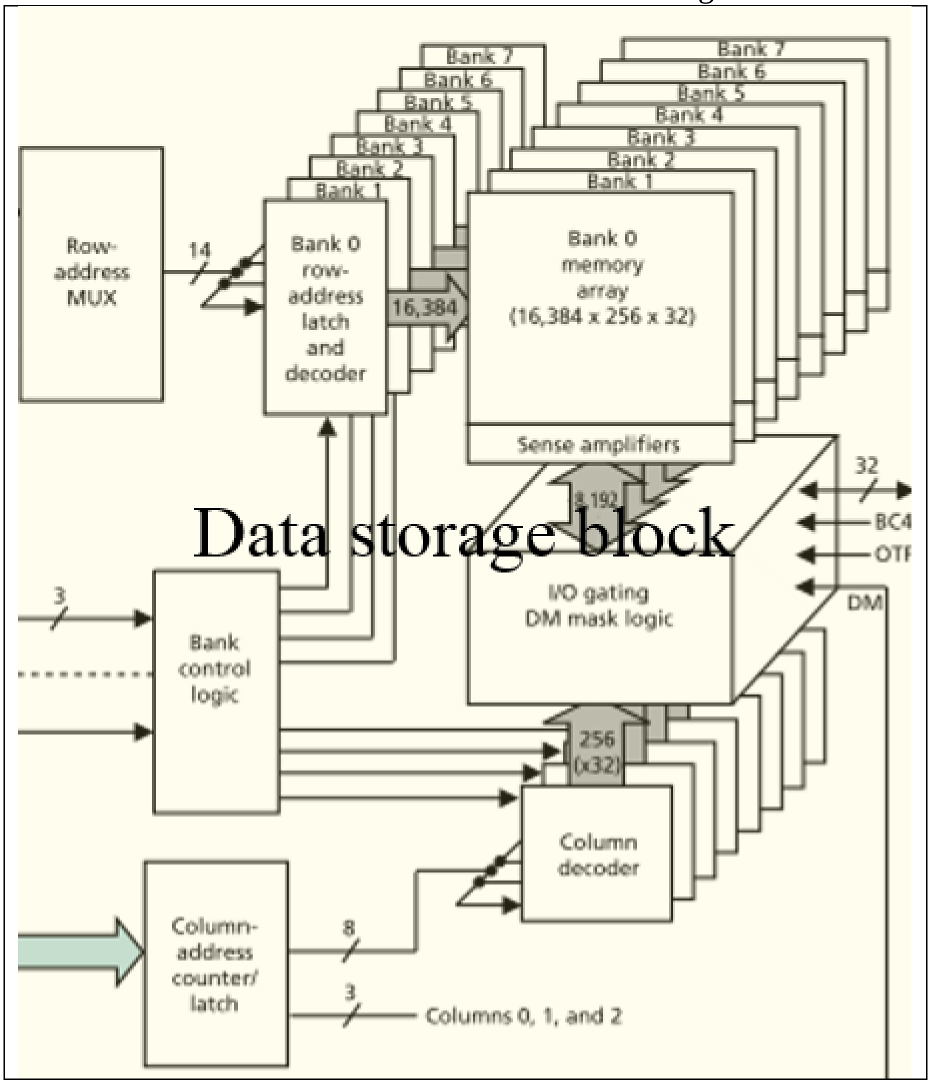


Figure 78: Data storage block substructure (from [42]).

This block is responsible

- In read mode: of the <u>delivery</u> of some data burst on the request of the Control and Addressing Block and with the starting address it received from this block;
- In write mode: of the <u>storage</u> of some data burst on the request of the Control and Addressing Block and with the starting address it received from this block.

8.5.3.2.2 Failure modes

8.5.3.2.2.1 Loss of message

A burst can be lost during its extraction from the bank. This may cause loss of data at memory level.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

A burst can be lost during its storage in the bank. This may cause, when corresponding zone will be read, the reading of aberrant data and thus the transition to valid erroneous data or to invalid data at processor level.

8.5.3.2.2.2 Untimely transfer of message

It could occur that this block untimely transfer data to the data transfer interface.

8.5.3.2.2.3 Abnormal sequence of messages

It could occur that the block mixes the burst so that some rows are not read or write in the right order. Such a failure would cause data stored or delivered to be wrong.

8.5.3.2.2.4 Untimely or forbidden transition of information

It could occur that the block corrupt some data, during the time it is stored or during storage or extraction operations, so that this data suffer an untimely transition to an incorrect valid value. We consider that the forbidden is not realized as a bit as always a value (0 or 1). So no transition to an invalid value can occur.

The first case would be the worst because it would lead to the usage of incorrect but valid data by the processor.

8.5.3.2.2.5 Impossible transition of information

The arbitrary separation of blocks of Figure 52 and Figure 53 led to situate the FIFOs in the Data Transfer Interface block. A behaviour that could stick a data at the same value is thus considered as unlikely to occur in the current block.

8.5.3.2.3 Failure mitigations

See paragraph 8.5.4.

8.5.3.3 Data Transfer Interface

8.5.3.3.1 Description

Data transfer interface block has two functions: firstly, it transfers the read data from Data Storage Block to DDR controller. Reversely, it transfers the written data from DDR controller to Data Storage Block.

This block, represented on Figure 79, concentrates the particularities of the DDR in general and of the DDR3 in particular:

- The clocks CK and CK# control the read and write operation and allows the double data rate;
- The prefetch buffers (2n for DDR, 4n for DDR2, 8n for DDR3) that allows “prepare” xn adjacent words for transfer.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

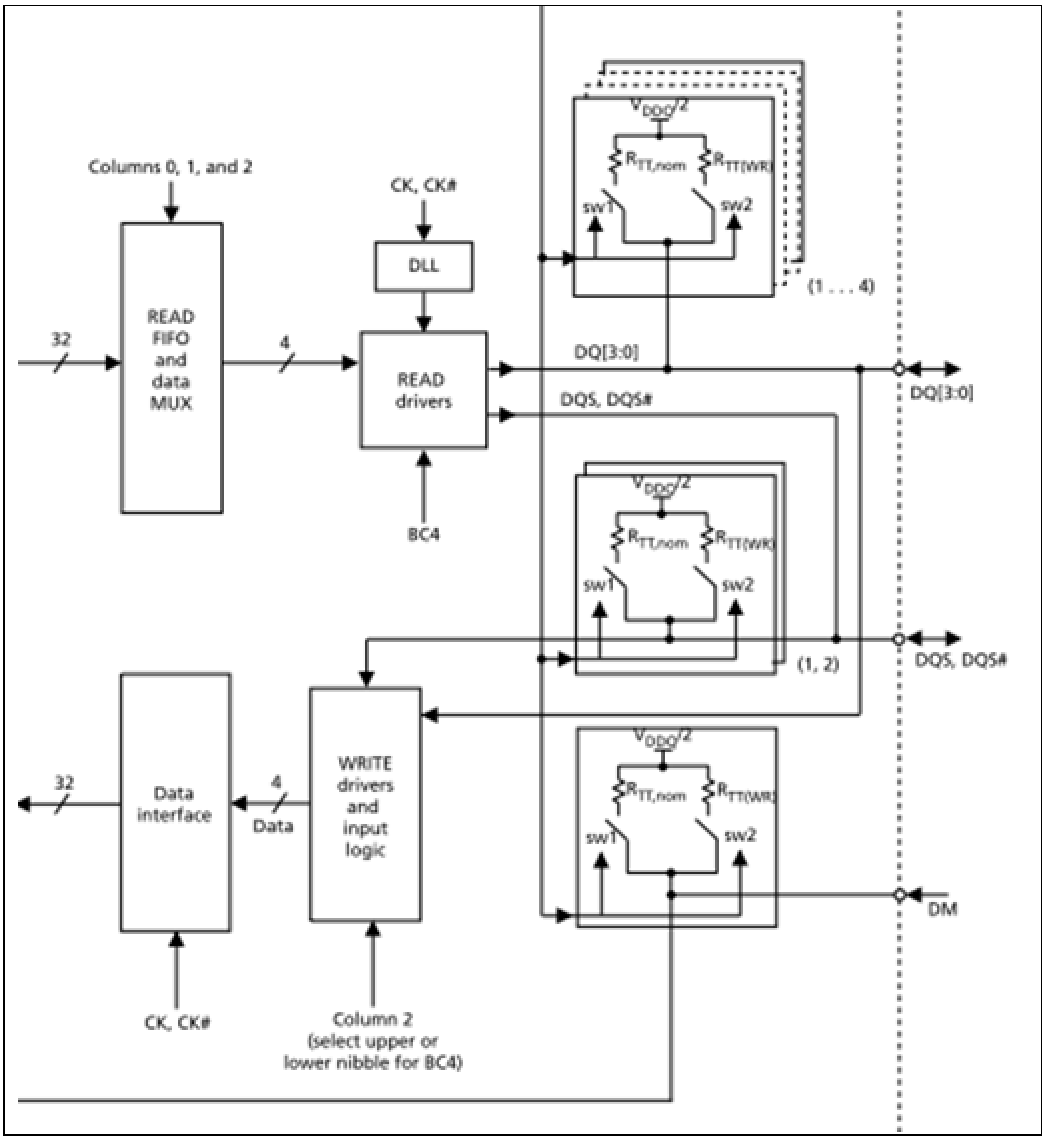


Figure 79: Data transfer interface block substructure (from [42]).

8.5.3.3.2 Failure modes

8.5.3.3.2.1 Loss of message

Loss of message (loss of some data in a burst) is an admissible mode of this Block. This could arrive for instance in case of prefetch buffer error, clock error or in the case of a write operation in case of untimely activation of DM signal.

8.5.3.3.2.2 Untimely transfer of message

It could be envisaged unless improbable that words stored in a buffer are untimely released. This should have not effects as either in read and write operation the receiver of the untimely transaction should not be ready to receive it.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

8.5.3.3.2.3 Abnormal sequence of messages

It seems very improbable that two parts of a burst are sent in an abnormal sequence.
Another abnormal sequence could arrive in case of collision between a "read" and a "write" burst. Such a collision should be avoided by the DDR controller, however in case of a delayed read operation in this block it could be imagine unless improbable that a write operation interferes with it. Such a delay of read operation could occur in case of transient error of the DLL that clocks the read drivers (Figure 79).

8.5.3.3.2.4 Untimely or forbidden transition of information

An untimely transition of information could rise during a burst transfer, in a buffer, even it is more current in the memory banks themselves (see paragraph 8.5.3.2).

8.5.3.3.2.5 Impossible transition of information

An impossible transition of information (bit stuck at some value) could occur in this block.

8.5.3.3.3 Failure mitigations

See paragraph 8.5.4.

8.5.4 Failure Detection & Mitigation

No failure mitigation is encountered at individual memory level.

8.5.5 Concluding remarks

Errors in DDRx SDRAM memories design result in failures on data provided to the microprocessor.
The DDRx chip in itself does not embed any local detection and mitigation mechanisms.

However, all electronic Memories are sensible to external aggression like atmospheric perturbation SEU or MBU that can change their content. This is why it is necessary to embed in DDR controllers, mechanisms able to detect such errors in particular in Avionic or Space environments. These mechanisms cover by the way most of the modes envisaged in the preceding paragraphs.

The two types of mechanisms implemented on memories are:
- Parity Checking,
- Error Code Corrector.

As already outlined theses mechanisms are not implemented on the memory chip but on an ensemble of memory chip driven by a DDR controller. They are described in chapter 9 with some others that may be implemented on memories (CRC, data mirroring).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.6 FLASH MEMORIES

8.6.1 Introduction and available data

As outlined in section 5.3.3, NAND flashes are of particular interest due to their complexity. Consequently examples considered in this section are of this type.

NAND flash Memory chip are principally described by a datasheet that covers the architecture, features, electrical connexion and environmental constraints data.

In the following paragraph we consider typical NAND flash memory and NAND flash card, which structures are deduced from several sources.

Document name	Document description	Examples
Datasheet	Datasheet for NAND Flash controllers	GreenLiant GLS55VD031: [44], GreenLiant GLS55VD020 [45] Lattice Semiconductor: RD1055 [46] QuickLogic [47]
Datasheet	Datasheet for NAND Flash card	Spansion [48]
Application Notes		Spansion [49]

Table 12: list of input in order to analyse a NAND Flash memory;

8.6.2 Architecture description

High capacity NAND Flash can be built by connecting in a same component several NAND Flash memory stacked dies like those represented on Figure 80.

Most of the considered NAND Flash devices rely on the Open NAND Flash Interface (ONFI) Specification for defining the host system interface.

An ONFI (Open NAND Flash Interface) command “Read Parameter Page” allows the host to access to standardized parameter fields that describe useful characteristics of the NAND Flash devices. The host takes advantage of the parameters read to adjust through its embedded firmware the fine management of the NAND Flash devices.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

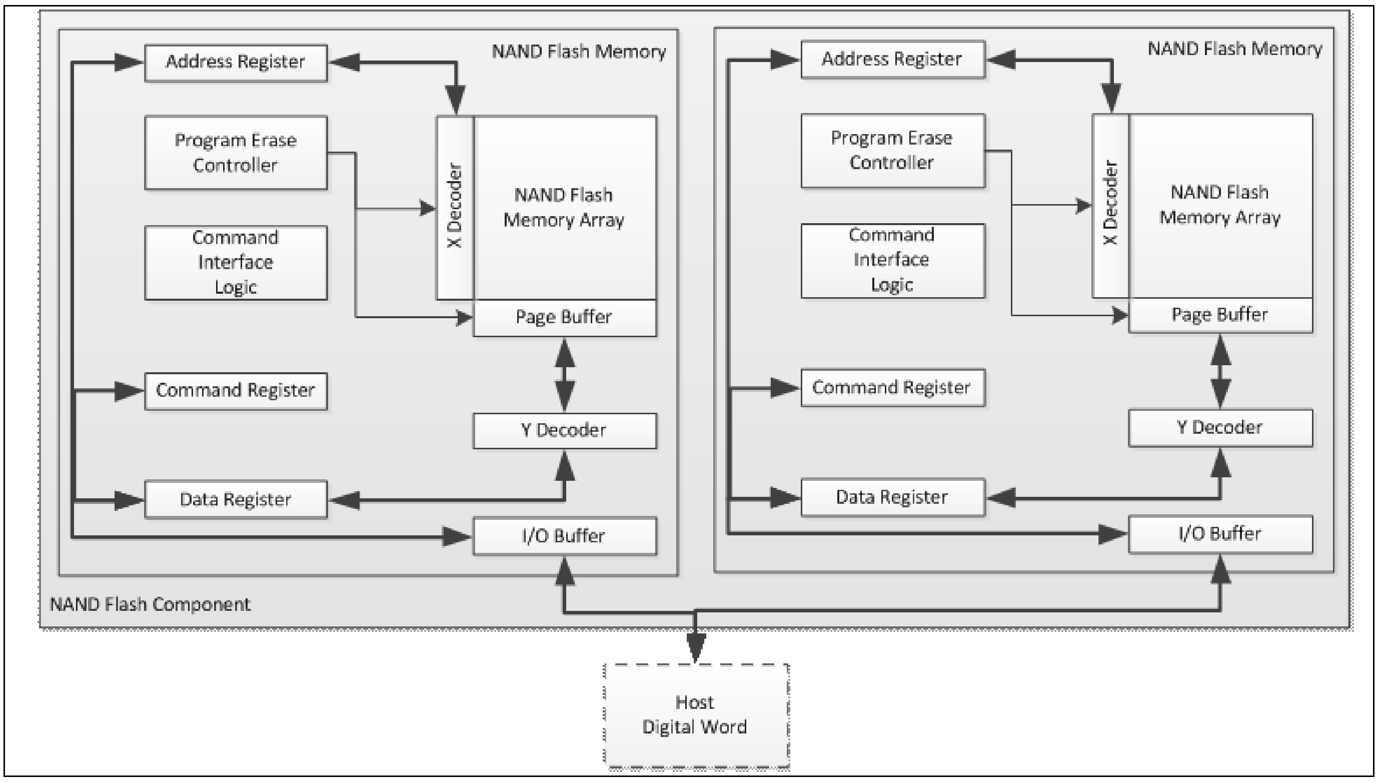


Figure 80 : Structure of a NAND Flash component

At this extend, solid-state drive appear as host subsystems (Figure 81).

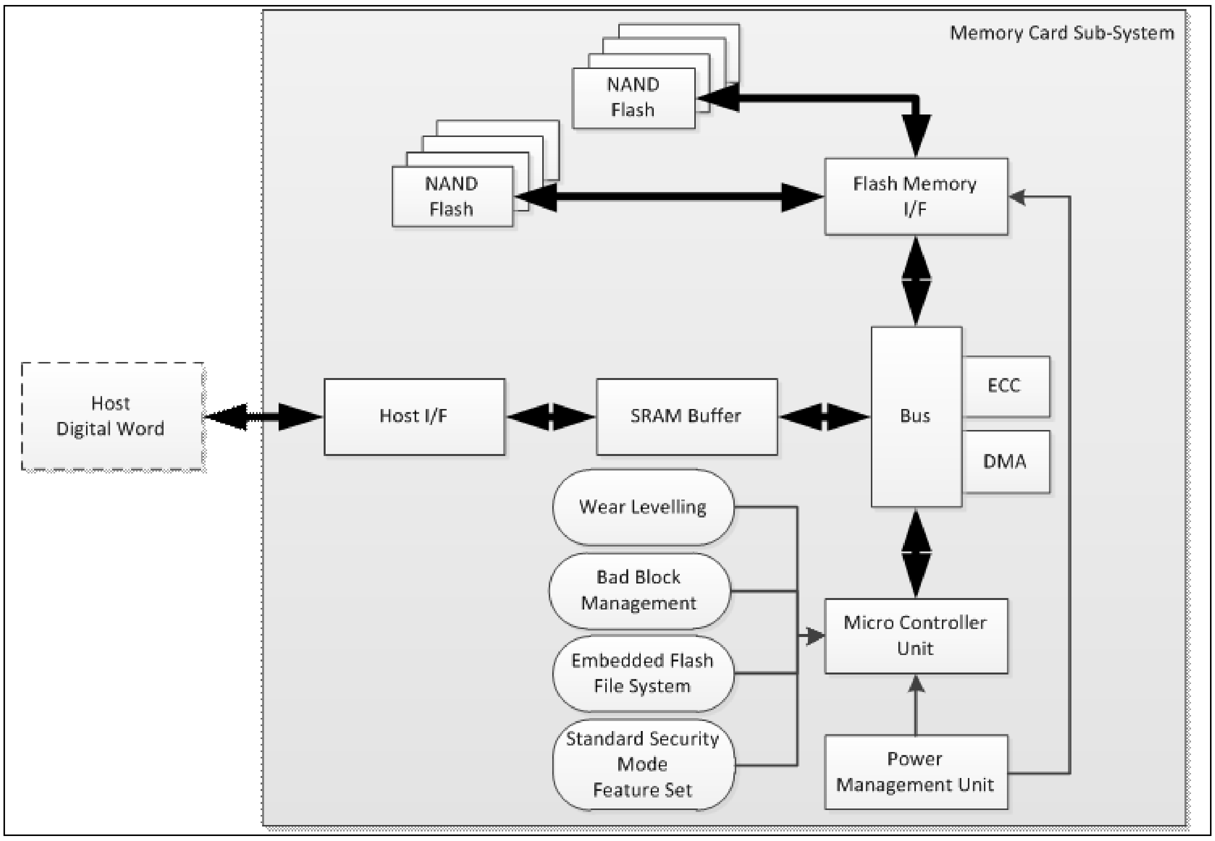


Figure 81 : Memory card structure

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

The sub-system embeds one of several NAND Flash dies and accesses to them through an interface (ONFI specification can be used).

On the other side an interface is proposed to the Host system, like:

- Buses: ATA, IDE, USB,
- Standardized interfaces: SD Memory Card, Multimedia Card…

SRAM Buffers are used to store data temporarily and give the necessary flexibility to adapt format of incoming and outgoing data streams.

Due to the admitted failures in the NAND Flash memory arrays, an ECC mechanism is required on the internal memory card bus (by the NAND Flash memory datasheet to ensure integrity performances). To ensure high transfer speed performances, DMA mechanism is embedded.

To provide functional higher level services, a Micro-Controller Unit is embedded in the sub-system.

The following functions are proposed and managed through the embedded firmware:

- Bad Block Management (the defective blocks identified during NAND Flash memory manufacturing are identified at sub-system level and their list managed to ensure integrity of the stored data),
- Wear levelling: the number of accesses to the NAND Flash memory has an impact on the stored data integrity and on the response time, so the firmware will manage the long lasting time and the number of block erase, the number of block write and even the number of read (read disturb) to provide the best service in term of integrity and longevity,
- Adaptation of the transferred data packets from/to the host system file to/from the NAND Flash memory block structures, with the maximum efficiency by using the proposed DMA mechanism,
- Security application can be hosted too to protect unauthorized access to stored data with key data encryption (e.g. standard ATA Security Mode feature set, Content Protection for Recordable Media copyright protection on SD Card, …)

A Power Management Unit controls the power consumption of the memory card subsystem and NAND Flash interface to avoid operation at risk on NAND Flash data, in particular during power transitions. The sub-system must be able to boot and recall data after unexpected power failures interrupt flash operations and controller built in power fault tolerance is expected.

8.6.3 Failure modes

8.6.3.1 Loss of message

In case of failure in the command interface logic, the data operation will not be effective and the data can be lost.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

8.6.3.2 Untimely transfer of messages

In case of failure in the command interface logic, an erroneous data operation can be performed and non-requested data can be proposed at the interface level.

8.6.3.3 Abnormal sequence of message

In case of corruption of the Read Parameter Page, the commands addressed to the Flash devices will not have a comprehensive sense and the realized sequence will be abnormal.

8.6.3.4 Untimely or forbidden transition of information

If the last program operation was interrupted before completion by power interrupt, even with a data verified correctly at the time of interruption, the page's data retention time will not reach its full potential. This will lead to un-correctable bit failures when the page is accessed later.

8.6.3.5 Impossible transition of information

In case of erroneous access to manufacture identified Bad Blocks, the data will not be stored in the NAND Flash memory.

8.6.4 Intrinsic robustness

No intrinsic robustness identified

8.6.5 Intrinsic failure mitigation mechanisms

The Read Parameter Page command can be considered as an embedded failure mitigation mechanism as it gives access to user on the warranted performances of the installed NAND Flash device. By using and checking this usage domain, the user will be able to monitor the component behaviour.

The embedded standardized identification of manufacture identified defective blocks is another means to help for failure mitigation.

NAND flash embeds an ECC realized by a hardware accelerator in order to correct the sector failures so that untimely or forbidden transition of data or impossible transition of data (address remained correct) are detected and corrected[46]. Critical data should be protected by complementary mechanisms directly by applicative command or by services offered by OS: e.g. CRC, Check sum or double copy of data in two different zones with complement to 1. Correspond to end to end between core and memory. These mechanisms are detailed in section **Erreur ! Source du renvoi introuvable.**.

8.6.6 Concluding remarks

Errors in Flash memories design results in failures on data stored and delivered. The mechanisms embedded on memory cards are not dedicated to cover design errors but manufacturing process dispersion

[46] The number of error detected and the number of error corrected depend upon the ECC implemented.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

and wearing. By the way they can cover such errors but could then be less efficient in the completion of their prime function. It seems preferable to test memory cards and to monitor (when possible) the triggering of embedded detection mechanism during the test (see subchapter 9.2) in order to guarantee the reliability of their design than relying on embedded mechanisms in order to guarantee the integrity and availability of their outputs. These embedded mechanisms should then be considered as complementary mechanisms for design errors and should be complemented by architectural mechanisms.

8.7 MICROCONTROLLERS

8.7.1 Introduction and available data

Microcontroller evolution has been described in section 5.3.4. This chapter considers Freescale microcontroller like for instance MPC8610 initially designed for embedded applications that process or display graphical images, such as robotics, in-vehicle infotainment, cockpit displays, single-board computers and multi-function printers and scanners.
This microprocessor implements the following features:

- e600 core built on Power Architecture technology with 256 KB backside L2 cache with ECC and integrated vector processing engine to accelerate image recognition and encoding/decoding (AltiVec®);
- DDR/DDR2 SDRAM memory controller with ECC (up to 533 MHz);
- Integrated display controller;
- Two PCI Express® Interfaces, one with 1x/2x/4x/8x lanes for connecting graphics processors;
- PCI 2.2 Interface at 32-bits and 66 MHz;
- Two four-channel DMA controllers;
- Enhanced local bus with 32-bit multiplexed address/data for ROM, NAND or NOR flash;
- I²S or AC97 audio inputs/outputs;
- Two fast/serial infrared interfaces (FIRI/SIRI);
- Serial peripheral interface (SPI);
- Two dual universal asynchronous receiver/transmitters (DUARTs);
- Two I²C controllers;
- Up to 32 general-purpose input/output (GPIO) ports.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

The basic documents available for the study of a microcontroller are listed in Table 13.

Document name	Document description	Examples
Product Fact sheet	Provides an overview of the microcontroller features as well as application use cases.	MPC8610 Fact sheet [50]
Datasheet	Includes all the hardware design related information as power supply specification timings, thermal environment.	MPC8610 Integrated Host Processor Hardware Specifications [51]
Reference Manual	Describes the features and operation of the microcontroller	MPC8610 Integrated Host Processor Reference Manual [52]
Core Reference Manual	Describes the features of the core	E600 Reference Manual [53]
Errata list	Details all known silicon errata on the component.	MPC 8610 Errata Sheet [54]
Application notes	Some application notes are emitted by the manufacturer in order to configure or use correctly the device. Often these application notes complement and particularise the reference manual and eliminate some ambiguities.	

Table 13: microcontroller list of reference documents

Some other sources may be available

- White papers, Publication and patents [55] emitted by the manufacturer have not the official character of a Reference Manual but can help to have an overview on a particular topic and help to ask relevant questions to the COTS manufacturer;
- Due to the complexity of the microprocessor and of its documentation (several thousands of pages) some training are proposed by the manufacturer and by affiliate consultants. These training cannot be considered as elements of proof but they can help a lot in the overall understanding of the COTS functioning;
- Data under No Disclosure Agreement are in general requested from the component manufacturer in order to improve and confirm the relevance of the functional description and of the failure behaviour.

8.7.2 Architecture description

The MPC8610 architectural description given in the open documentation is reported on Figure 82.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

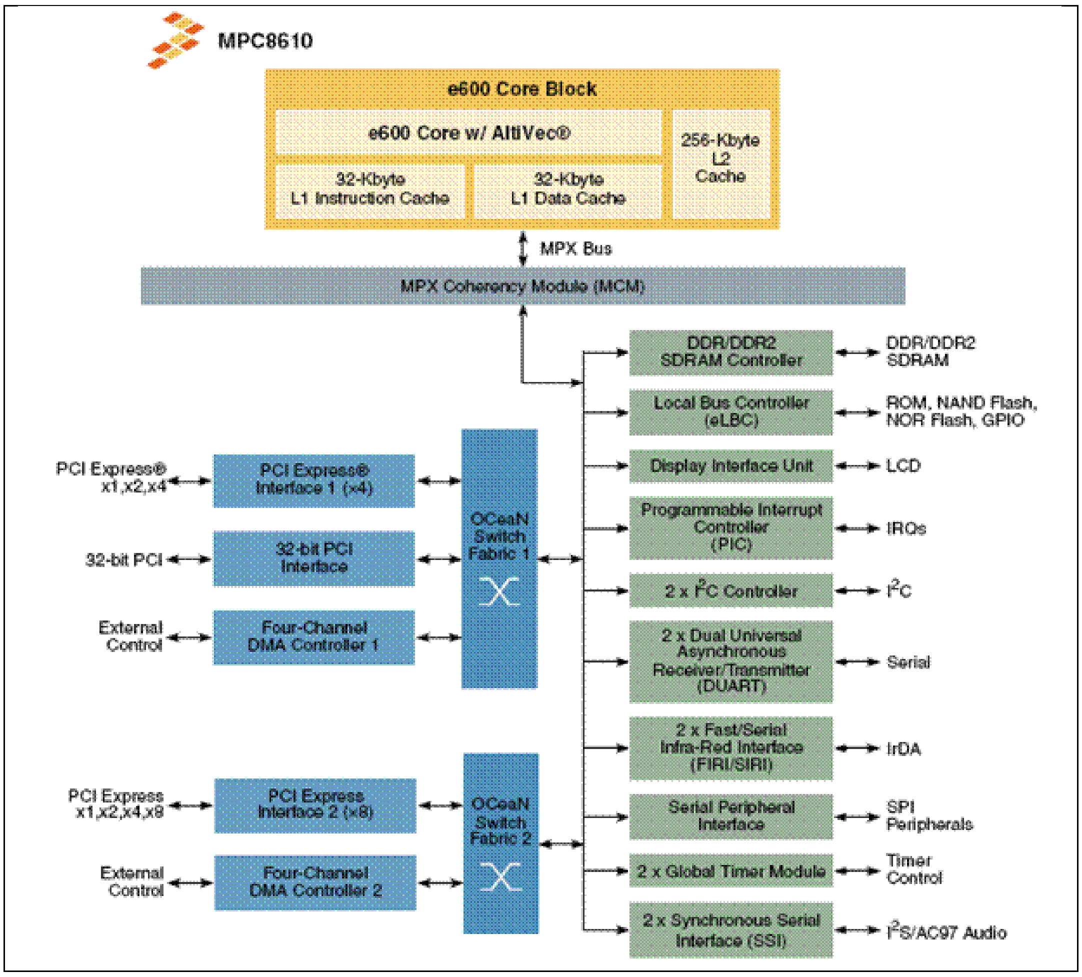

MPC8610
e600 Core Block
e600 Core w/ AltiVec®
32-Kbyte L1 Instruction Cache
32-Kbyte L1 Data Cache
256-Kbyte L2 Cache
MPX Bus
MPX Coherency Module (MCM)
DDR/DDR2 SDRAM Controller
DDR/DDR2 SDRAM
Local Bus Controller (eLBC)
ROM, NAND Flash, NOR Flash, GPIO
Display Interface Unit
LCD
Programmable Interrupt Controller (PIC)
IRQs
2 x I²C Controller
I²C
2 x Dual Universal Asynchronous Receiver/Transmitter (DUART)
Serial
2 x Fast/Serial Infra-Red Interface (FIRI/SIRI)
IrDA
Serial Peripheral Interface
SPI Peripherals
2 x Global Timer Module
Timer Control
2 x Synchronous Serial Interface (SSI)
I²S/AC97 Audio
PCI Express® x1,x2,x4
PCI Express® Interface 1 (x4)
32-bit PCI
32-bit PCI Interface
External Control
Four-Channel DMA Controller 1
OCeaN Switch Fabric 1
PCI Express x1,x2,x4,x8
PCI Express Interface 2 (x8)
External Control
Four-Channel DMA Controller 2
OCeaN Switch Fabric 2

Figure 82 : MPC8610 block diagram from [52].

This block diagram exhibit the features already listed. It is important to note that this microcontroller is architected around a bus (MPX bus) with an arbiter (MPX Coherency Module – MCM). This Microcontroller is thus of the type described on Figure 13. Microcontroller architecture around interconnect is detailed in next subchapter.

8.7.3 Block study

8.7.3.1 Core

8.7.3.1.1 Description

MPC8610 core (e600) is described in the core reference manual [53] and in Part II of MPC8610 Reference Manual [52]. E600 core has:

- L1 cache for instructions (32kB) and L1 cache for data (32kB);
- Mixed data and instructions L2 cache (256kB).

E600 is a 32 bits core implementing several levels of pipelining and different levels of buffer allowing fetching instructions, facilitating branch prediction, queuing instructions, etc. (see Figure 83). Contrary to e500mc presented on Figure 89, e600 has a module of vectorial computation based on AltiVec®

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Technology. This vectorial computation capability and the MMU structure are the greatest difference with the e500mc that will be developed on paragraph 8.8.3.1.

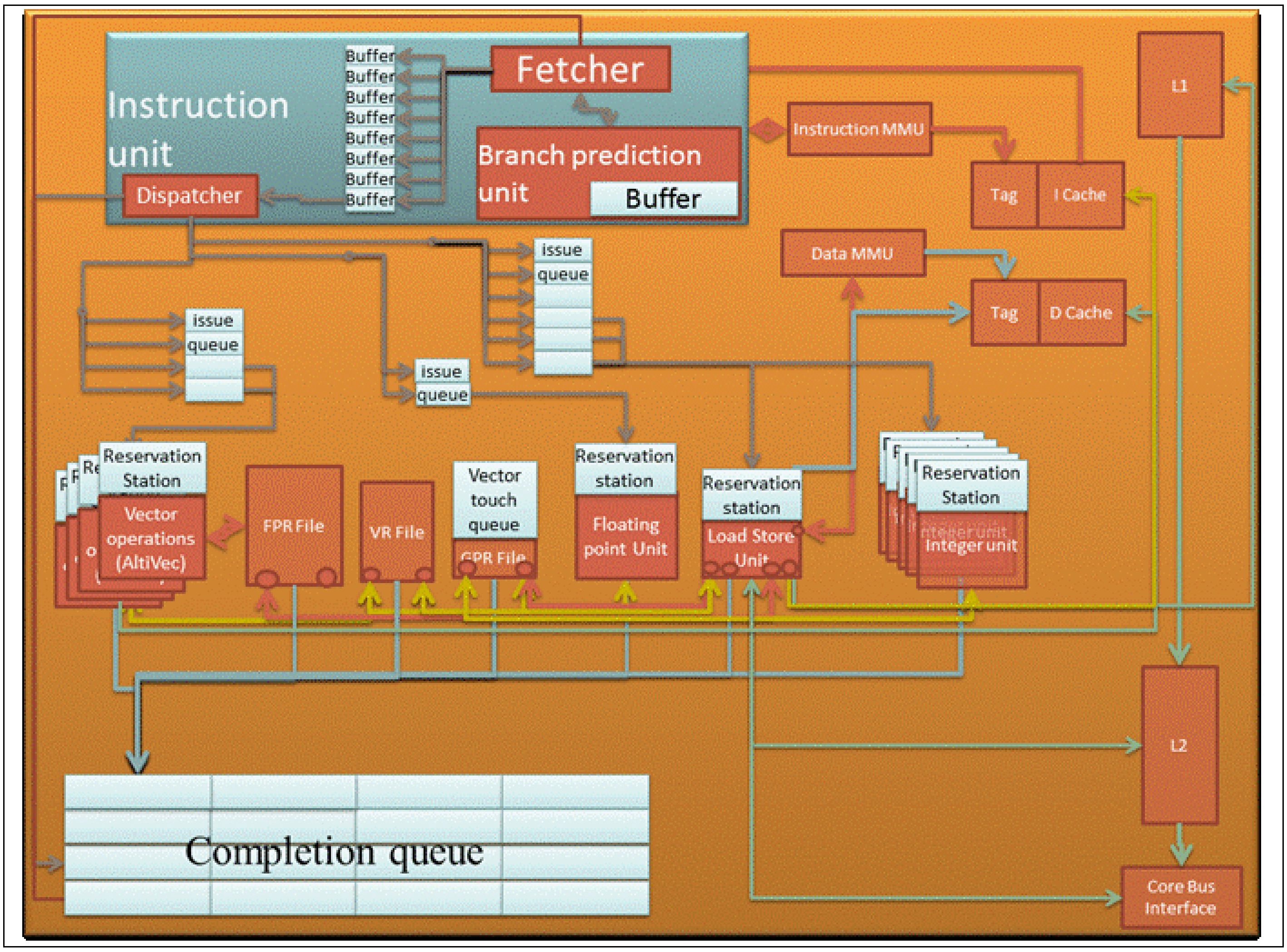


Figure 83: Simplified e600 block diagram focusing on instruction paths and memory type areas (clear grey zones) – adapted from [53];

8.7.3.1.2 Failure modes

Failure modes of the core are those already listed in the section 6.3 for the interface between hardware and software:

- No program instruction outing, for instance if instruction is lost in buffer or queue;
- Erroneous calculation outing, for instance if completion queue Figure 83 that stores results waiting the completion of pipelined instruction is corrupted;
- Latency in program instruction outing (Maximum Execution Time drift), for instance if branch prediction unit (Figure 83) is erroneous and in certain conditions take systematically wrong assumption;
- Inversion of tasks, if a problem occurs in the filling of completion queue.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

Toward its interface with MPX bus, each core can develop the standard failure modes developed in the first part of section 6.3 and already experienced on COTS interfaces in chapter 7.

8.7.3.1.2.1 Loss of message

Due to possible conflict in cache synchronization processes data or addresses could be lost before their transfer to the internal bus. It is probable that such behaviour necessitate a second error in the information transfer mechanisms.
A more probable mechanism can be generated by core interruption on core external request or on core self-request (see following section).

8.7.3.1.2.2 Untimely transfer of message

Because of interface buffers it is possible that core emit repeatedly the same data / address set to internal bus and occupy it or occupy the resource to which it sends this set.

8.7.3.1.2.3 Abnormal sequence of messages

Messages transfer between Cache and internal bus could develop such behaviour for instance if cache descriptors are corrupted. The involvement of buffers between L2 caches and internal bus cannot directly lead to these failure since they seems to be of FIFO type. However when buffers are involved, the possibility of data output inversion in the stack because of some pointer error can be envisaged.

8.7.3.1.2.4 Untimely or forbidden transition of information

Data and addresses can be corrupted in the core buffers or in the cache or in intermediate buffers between cache and internal bus. Indeed e600 cores implement different level of pre and post cache buffers (not shown on Figure 83) allowing optimisation of cache operations. Contrary to the caches, these buffers are not protected by ECC or parity mechanisms (see paragraph 8.7.3.1.3). The effect of such failure is identical as the Erroneous calculation outing.

8.7.3.1.2.5 Impossible transition of information

See “untimely or forbidden transition of information”

8.7.3.1.3 Failure mitigation mechanisms

E600 core is partially covered by several mechanisms:

- Parity on L1 cache,
- Parity and ECC on L2 cache,
- Memory Management Unit - see paragraph 8.8.3.1.3.3.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.7.3.1.3.1 Parity on L1 cache

From the point of view of failure mitigation mechanisms, L1 cache can be separated into:

- L1 tags for instruction,
- L1 tag for data,
- L1 queues,
- Instructions,
- Data.

Due to the relative volume occupied by each zone of L1 cache, the level of protection defined by the manufacturer differs from zone to zone (Table 14: Mechanisms implemented on the L1 cache of e600).

Cache zone	Volume	Protection implemented
L1 tags for instruction	128 sets of 8 blocks of 24+1 bits	None
L1 tag for data	128 sets of 8 blocks of 24+3 bits	None
L1 queues	• 5 entries for load misses • 2 entries for instruction fetches • 2 for cacheable store requests • 2 LLQ (L1 Load Queue) • 3 LSQ (L1 Store Queue)	None
Instructions	32 KB + 1b/Word parity	Parity (1b / word)
Data	32 KB + 1b/B parity	Parity (1b / Byte)

Table 14: Mechanisms implemented on the L1 cache of e600;

8.7.3.1.3.2 Parity and ECC on L2 cache

L2 cache can be separated into:

- L2 tags,
- L2 data,
- L2 queues.

Different protection levels have been defined for each zone Table 15.

Cache zone	Volume	Protection implemented
L2 tags	512 sets of 8 blocks of 24+1(parity)+2(status)	Parity
L2 data	256KB +1b/B for ECC/parity	ECC + Parity (1b/B)
L2 queues	• Prefetch (3) , • L2SQ (L2 Store Queue) - 4 entries for L1 Castouts - 1 entry for snoop / push interventions	None

Table 15: Mechanisms implemented on the L2 cache of e600;

Protections on L1 and L2 caches should be compared to the protections implemented on e500mc on L1, L2, TLB, PAMU caches, etc. (see paragraph 8.8.3.1.3). This shows the continuous improvement made by

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

component manufacturers for the protection mechanisms embedded in their devices. This is necessary due to the increasing memory quantity embedded in a single device.

8.7.3.2 MPX bus and MPX coherency module

8.7.3.2.1 Description

The MPX Bus is a high-performance bus with separate address and data buses [56], each with its own set of arbitration and control signals (Figure 84). This allows for the decoupling of the data tenure from the address tenure of a transaction and provides for a wide range of system bus implementations, including:

- Non-pipelined bus operation,
- Pipelined bus operation,
- Split transaction operation.

.

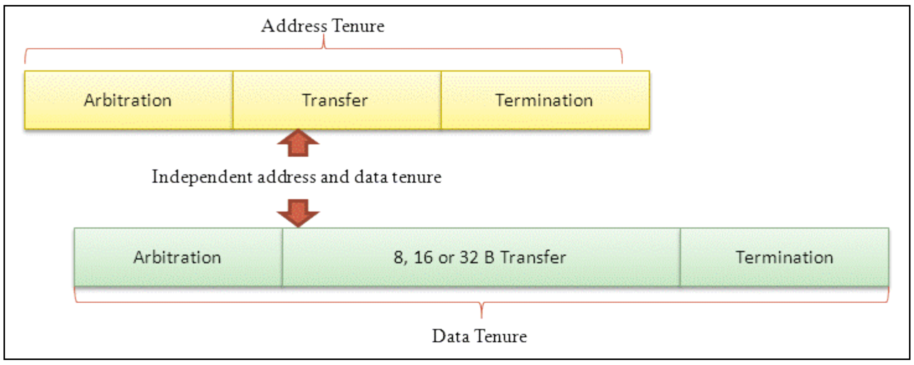


Figure 84: MPX bus data and address tenure

Arbitration for both address and data bus mastership is performed by an external arbiter (located in the MPX Coherency Module) using the address arbitration signals BR (Bus Request), and BG (Bus Grant), ARTRY (Address Retry), DRTRY (Data Retry) and the data arbitration signal DBG (Data Bus Grant). Most arbiter implementations require additional signals to coordinate bus master, slave, and snooping activities.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

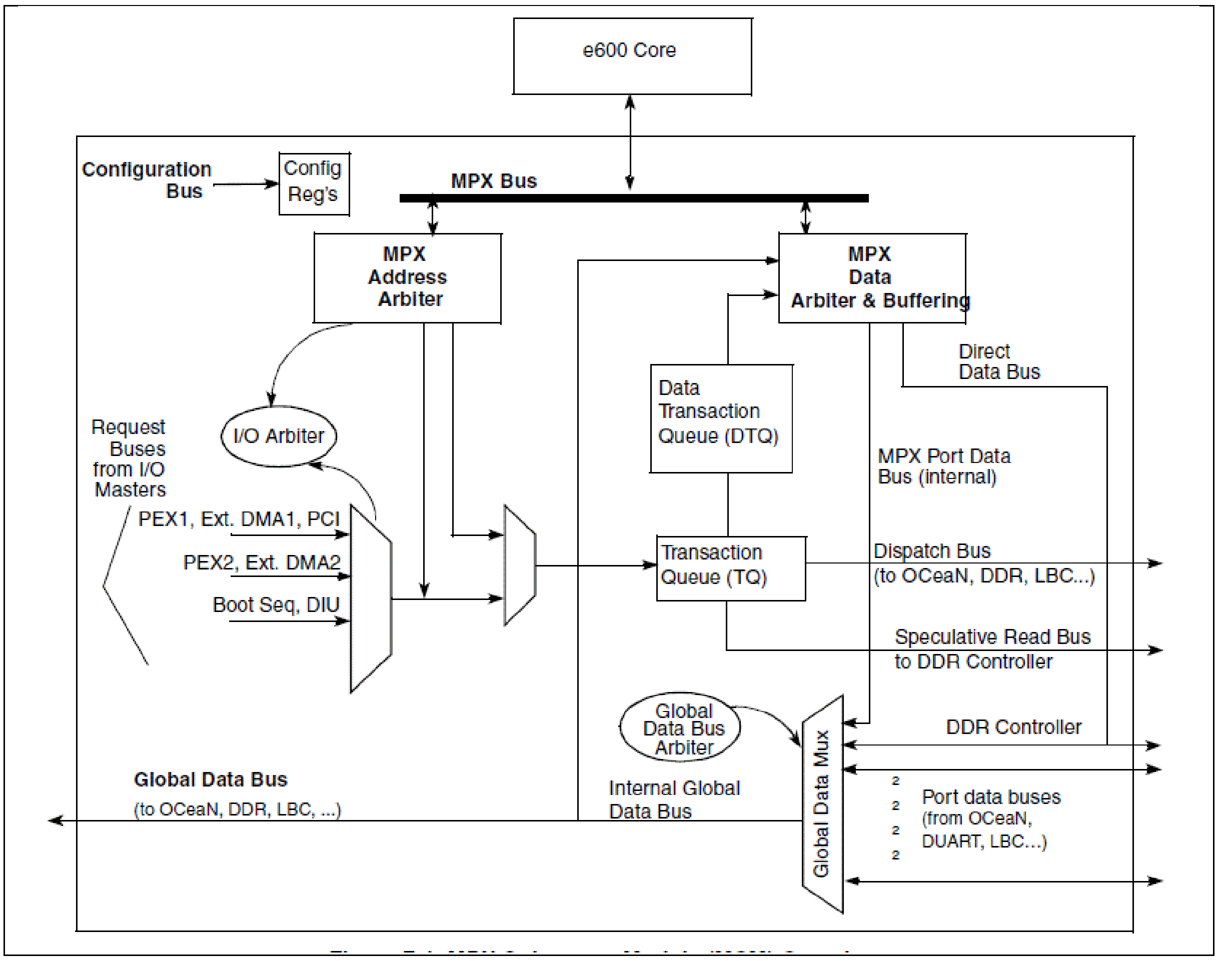

e600 Core
Configuration Bus
Config Reg's
MPX Bus
MPX Address Arbiter
MPX Data Arbiter & Buffering
Direct Data Bus
Data Transaction Queue (DTQ)
MPX Port Data Bus (internal)
Request Buses from I/O Masters
I/O Arbiter
PEX1, Ext. DMA1, PCI
PEX2, Ext. DMA2
Boot Seq, DIU
Transaction Queue (TQ)
Dispatch Bus (to OCeaN, DDR, LBC...)
Speculative Read Bus to DDR Controller
Global Data Bus Arbiter
DDR Controller
Global Data Bus (to OCeaN, DDR, LBC, ...)
Internal Global Data Bus
Global Data Mux
Port data buses (from OCeaN, DUART, LBC...)

Figure 85: MPX Coherency Module

MPX bus transfers instruction as data, some of them are directly interpreted in particular in order to acts on the caches..

8.7.3.2.2 Failure modes

8.7.3.2.2.1 Loss of message

At the interface between core and the bus (system bus interface) some buffers (load queue, Bus Store Queue, Castout queue, push queue, can take place that could suffer a loss of transactions.
The same behaviour could be exhibited by MCM (see Figure 85).
In case of error of the bus arbiter in the MPX coherency Module, Address and data separation could cause loss of one or both.

8.7.3.2.2.2 Untimely transfer of message

Again, because of queues and buffers, MPX could either generate untimely transfer of message with delay and with repetition (babbling). See chapter on bridges for a detailed discussion of possible discrepancies between buffers and queues.

8.7.3.2.2.3 Abnormal sequence of messages

Abnormal sequence of message could be generated by an error of pointer.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE TECHNICAL LIBRARY ABBOTTAEROSPACE.COM

8.7.3.2.2.4 Untimely or forbidden transition of information

Due to buffer management and to high bandwidth transfers on MPX, data can suffer from untimely transition. Addresses can suffer of untimely transition or of forbidden transition if the new destination address does not exist. In case of instruction transfer, untimely transition to some operations can realize operation on the caches that could invalidate the complete application run.

8.7.3.2.2.5 Impossible transition of information

Because of buffer management impossible transition could occur.

8.7.3.2.3 Failure mitigation mechanisms

Failure mitigation mechanisms implemented by the MPX bus are the following:
- An address and a data parity signals are sent in parallel to address / data transfers.
- In address termination phase an acknowledgment is sent with a signal for completed address tenure or retry request
- In data termination phase, an acknowledgement is requested after each beat of 8 Bytes (data termination signal). A special data termination signals (final data beat) is sent at the end of the burst.

8.7.3.3 DDRx Controllers

8.7.3.3.1 Description

DDR2 controller is described in chapter 8 of MPC8610RM [52]. The following analysis covers also DDR3 controllers of multicore (see for instance chapter 11 of P4080RM [33]).
The MPC8610 DDR memory controllers support DDR2 SDRAM. The memory interface controls main memory accesses. The memory controller also supports chip-select interleaving within a controller as well as interleaving across controllers on bank, page, or cache line boundaries. The MPC8610 can be configured to retain the currently active SDRAM page for pipelined burst accesses.
Page mode support of up to 32 simultaneously open pages can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save 3 to 4 clock cycles from subsequent burst accesses that hit in an active page.

The following Figure 86 shows the internal block diagram to the MPC8610 DDR controller.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

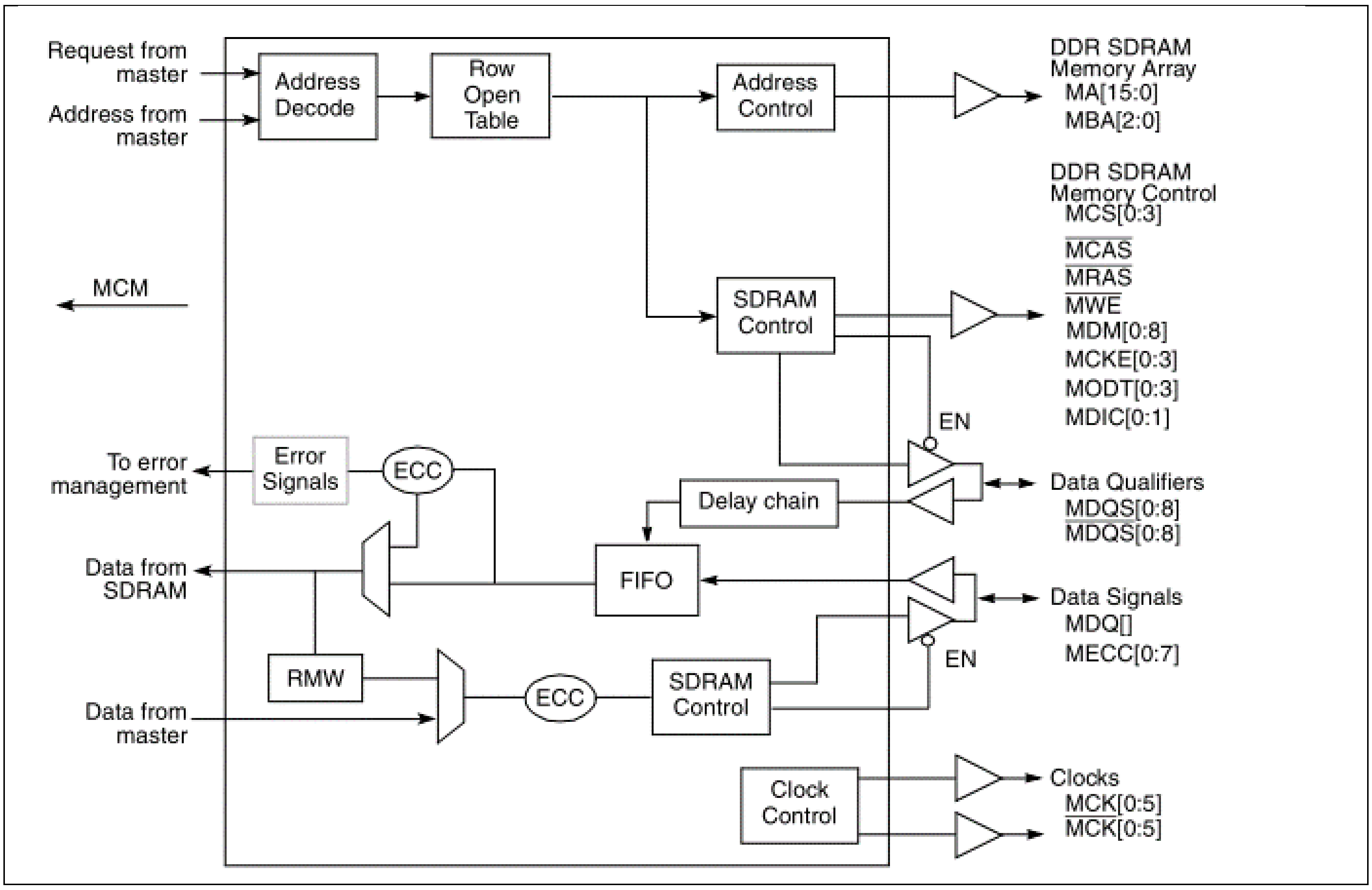


Figure 86: DDR controller internal block diagram [52].

One can visualize on this diagram the ECC encoder and decoder and some buffers type memories (FIFO and Delay chain) introduced in reading chain.

8.7.3.3.2 Failure modes

8.7.3.3.2.1 Loss of message

Due to its design DDR controllers can loss part of a burst in reading.
From this point of view, the transfer path in writing is more direct and loss of data on this path is improbable.
An error in ECC encoder/decoder configuration or design could also make burst of part of burst to be rejected and be considered as lost. This will again be effective in reading even if the error occurs during writing.

8.7.3.3.2.2 Untimely transfer of messages

As they store information in buffer it is possible to occupy the memory interface bus with a repetition of the same data. So, again in reading, it should be possible to imagine such behaviour. It requests an error in the SDRAM controller of Figure 86.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

8.7.3.3.2.3 Abnormal sequence of messages

Due to internal arbitration on information in buffers and possible corruption of configuration registers of DDR controllers, data can be read in memory in an abnormal sequence.

8.7.3.3.2.4 Untimely or forbidden transition of information

Data read or written can be corrupted during transfer or during manipulation by the controller.

8.7.3.3.2.5 Impossible transition of information

See “untimely or forbidden transition of information”.

8.7.3.3.3 Failure mitigation mechanisms

An ECC detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble (half a Byte).
Upon detection of a loss of power signal from external logic, the DDR controllers can put compliant DDR SDRAM DIMMs into self-refresh mode, allowing systems to implement battery-backed main memory protection.
In addition, the DDR controllers offer an initialization bypass feature for use by system designers to prevent re-initialization of main memory during system power-on after an abnormal shutdown.

8.7.3.4 OCeaN^{TM}

8.7.3.4.1 Description

High-speed peripheral interfaces PCI, PCIE and SRIO, connect to a common crossbar switch referred to as OCeaN^{TM}. As mentioned within [13] information about this crossbar is difficult to obtain and in particular reference manual is very elusive on this topic. It can be useful to consider various other sources and in particular patents like [55].

OCeaN^{TM} is connected to

- PCIe controller
- SRIO controller
- PCI controller
- DMA on one side, and
- MPX bus on the other side.

Note that in the case of multicore (see next subchapter 8.8), OCeaN^{TM} is no longer interfaced to PCI drivers but to more PCIe drivers and not to MPX bus but to the Microcontroller internal bus. Except this connection differences it seems that both OCeaN^{TM} has the same design.

8.7.3.4.2 Failure modes

8.7.3.4.2.1 Loss of message

OCeaN^{TM} could contain buffer switches (as indicated in [55]). Such a structure can exhibit a loss of message.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

8.7.3.4.2.2 Untimely transfer of messages

OCeaN^{TM} could contain a fabric Arbiter and a Fabric Master Controller (as indicated in [55]). Such structures could transfer the same message repeatedly and then degrade the communication amongst the PCI, PCIe, SRIO, DMA and MPX.

8.7.3.4.2.3 Abnormal sequence

Based on [55], OCeaN^{TM} could invert to information, due to controller error or buffer bit stuck at some value. Deadlock avoidance mechanisms could also create such behaviour.

8.7.3.4.2.4 Untimely or forbidden transition of information

Based on [55], OCeaN^{TM} could flip information (address, data or ctrl), due to controller error or buffer bit flip, or stuck information, due to controller error or buffer bit stuck at some value.

8.7.3.4.2.5 Impossible transition of information

Based on [55], OCeaN^{TM} could stick information, due to controller error or buffer bit stuck at some value. This could affect data as well as addresses.

8.7.3.4.3 Failure mitigation mechanisms

- The fabric arbiter embedded in version of OCeaN^{TM} described in [55] allows a packet transfer only if the destination can accept the packet. This could avoid, if effectively realized, a large part of the causes for "Loss of message". It ensures also that no deadlock situation can occur in the simultaneous processing of two high priority tasks.

- In the case of Multicores, IOMMU (for instance PAMU), also clearly not located in OCeaN^{TM} but in the paths between OCeaN^{TM} and CoreNet ^{TM};
- can stop major untimely or forbidden transition on address (see §8.8.3.3).

8.7.3.5 PCIe

8.7.3.5.1 Description

PCIe interface Controller is described in chapter 21 of MPC8610RM [52] and in the case of multicore in chapter 18 of P4080RM [33].
MPC8610 has 2 PCIe ports on connected to two different OCeaN.
PCIe has been described in subchapter 7.7. It is important to appreciate the MPC8610 PCIe controllers as the "Root complex" of Figure 44: PCIe bus topology on page 94.

Each of the PCIe Controllers is configurable as a PCIe Root Complex (see section 7.7.1) or a PCIe endpoint.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.7.3.5.2 Failure modes

The failures modes of PCIe have been described in subchapter7.7.

Failure modes of the PCIe Controller that can disturb several PCIe output are those, described in section 7.7.2 that are generated by high level layer (DLL or TL).

To these modes one can add loss or corruption of the configuration registers:

- Configuration as End Point or root complex:
    - As an initiator, the PCI Express controller supports memory read and write operations;
    - In addition, in RC mode, PCIe support configuration and I/O transactions;
    - As a target interface, the PCI Express controller accepts read and writes operations to local memory space.
    - When configured as an EP device, the PCI Express controller accepts configuration transactions to the internal PCI Express configuration registers.
    - Message generation and acceptance are supported in both RC and EP modes.
    - Locked transactions and inbound I/O transactions are not supported.

Corruption of this configuration does not lead to loss of the bus as it is coded on one bit and as the two values ensure normal communication.

- SerDes Protocol determines the link width;
- SerDes clock ratio and SerDes clock divider determine the link speed
  Corruption of these two configurations that can lead to a loss of the bus (change from PCIe to SRIO for instance) or to degradation of its performances.

8.7.3.5.3 Failure mitigation mechanisms

See subchapter 7.7.

8.7.3.6 DMA

8.7.3.6.1 Description

DMA controllers are complex IP developed by chip manufacturer to reduce the charge of the processor in memory to memory transfer. The DMA controller transfers blocks of data between the many interface and functional modules of the chip, with limited use of resources of the cores or external hosts (for instance PCIe when acting as master). Both the cores and external devices can initiate DMA transfers. The considered microcontrollers implement in general 2 DMA controllers with each 4 channels. Each channel is capable of complex data movement and advanced transaction chaining.
DMA organize input data into packets in order to transfer them optimally. The acting algorithms are complex and may generate by themselves some delays that should be taken into account in WCET estimation unless they are mastered.

Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and writes data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Typical DMA transfers are:

- From DDRx interface to DDRx interface,
- From DDRx interface to PCI, and from PCI to DDRx interface,
- From DDRx interface to PCIe, and from PCIe to DDRx interface.

Figure 87 presents the DMA controller internal structure from [52].

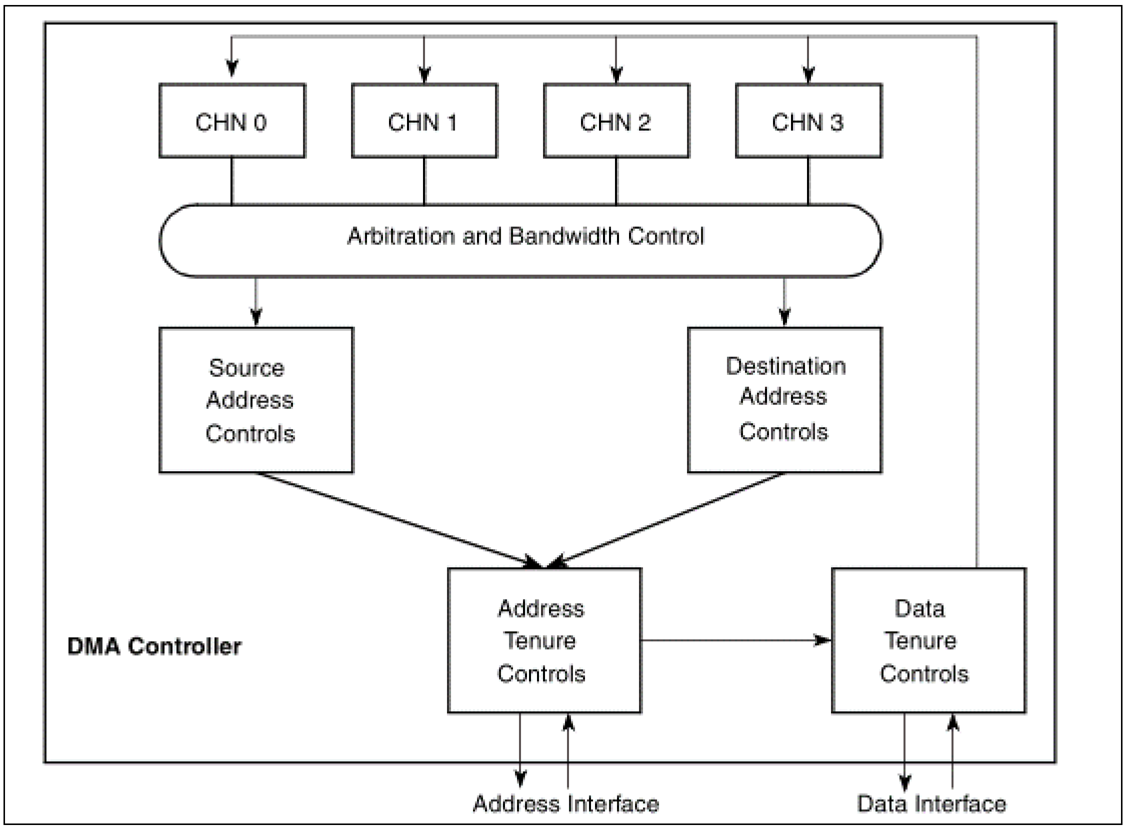


Figure 87: DMA controller internal block diagram [52].

8.7.3.6.2 Failure modes

MPC8610 internal DMA controllers use physical address and bypass the monitoring of accesses performed by the MMU. Therefore misbehaviour of the DMA due to a hardware design bug can lead to a transfer to wrong addresses by the DMA . Use of IOMMU on multicores (e.g. PAMU on P4080) mitigates this issue.

8.7.3.6.2.1 Loss of message

Loss of transaction by a DMA can be caused by

- Loss of address, either source or destination;
- Loss of data during the transfer by the DMA.

This loss can be total or partial. It cannot be excluded a priori that a DMA executes only a part of a transfer.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

8.7.3.6.2.2 Untimely transfer of message

DMA has to be considered as a transaction initiator within the micro controller. It cannot thus be excluded that due to a design error, it initiates transactions even if they are not requested.
It is also possible to imagine that it duplicates on a third target, a transaction initiated from one target to another. A particular error of this type should be a modification of CCSR (Configuration, Control, and Status Registers) by a DMA erroneous transfer.

8.7.3.6.2.3 Abnormal sequence of messages

Due to multiple channel structure, DMA could invert two transactions.

8.7.3.6.2.4 Untimely or forbidden transition of information

Due to some error in “data tenure control” DMA could generate some bit flip and thus some untimely transition of information.
In case of error on addresses, data could be sent to the wrong address. For valid addresses of destination, the result would be an untimely transition of data. For non-valid addresses (forbidden transition), the result should be a loss of data or more probably an impossible transition of information.

8.7.3.6.2.5 Impossible transition of information

Due to some error in “data tenure control” DMA could generate some bit stuck and thus some impossible transition of information. In this case the same data could be copied in various memory zones.

8.7.3.6.3 Failure mitigation mechanisms

No failure mitigation mechanisms are integrated to DMA.
On MPC8610, internal DMA controllers use physical address and bypass the monitoring of accesses performed by the MMU.
In the case of multicores, DMA failure modes related to untimely or forbidden transition of addresses are partially covered by the IOMMU (subsection 8.8.3.3).

8.7.3.7 General Purpose I/O driver

GPIO driver is covered through the discrete I/O interface description of subchapter 7.2.

8.7.3.8 SPI driver

SPI driver is covered through the SPI interface description of subchapter 7.3. It is considered here that others simple interface such as DUART and I²C exhibit similar possible behaviours.

8.7.3.9 CCSR (Configuration, Control, and Status Registers)

Configuration registers are treated in the multicore section. The challenges and results are comparable for a single core.

This Document Provided by the Abbott Aerospace
Technical Library
ABBOTTAEROSPACE.COM

8.7.4 Concluding remarks

Microcontrollers are complex aggregate of IP blocks that can be themselves complex. Errors of these IP blocks results in failures computational errors by cores and output message failures on some of the multiple output of the MCU: GPIO, SPI, PCI, PCIe, etc. Some IP Blocks embed detection and/or mitigation mechanisms that cover failures from the considered block and from some other blocks in interaction. These mechanisms are synthetized in section 9.3.4.1. They can help ensuring the detection and mitigation of microcontrollers' errors but for such complex COTS, mixed internal and architectural and full architectural means are necessary. The examples of detection – mitigation mechanisms given in sections 9.3.3 and following are in general applicable to MCU.

8.8 MULTICORE MICROCONTROLLERS

8.8.1 Introduction and available data

The points covered in this chapter are applicable to multicore version of Freescale P2, P3 last series[47], P4, P5 series (left side of Table 16). They will in most part remain true for future T series (left side of Table 16).

Family	2 cores	4 cores	8 cores	12 cores	Cores	L1	L2	L3
P2		P2040/ P2041			e500mc	Per core	None[48]	1 shared
P3		P3041			e500mc	Per core	Per core	1 shared
P4		P4040	P4080		e500mc	Per core	Per core	2 shared
P5	P5020	P5040			e5500	Per core	Per core	2 shared
T1	T1020	T1040			e5500	Per core	Per core	1 shared
T2		T2080			e6500	Per core	1 shared	1 shared
T4			T4160	T4240	e6500	Per core	Per cluster of 4 cores	As many as cluster quantity

Table 16: Examples of Freescale multicores microcontrollers sorted by families (left column) and documented with some of their features (right side);

These families are characterized by their implementation of

- PowerPC (QORIQ ™) cores with attached L1 and in general L2 caches per cores (see Table 16),
- A Switches Interconnect called CoreNet™ Coherency Fabric (CCF),

[47] This is true only for some versions of P2 family that implement interconnect.
[48] In other series L3 interfaces interconnect (or bus) with the DDR driver, so we consider arbitrary that P2 series do not have L2 cache but a L3 cache.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- Some large bandwidth IO,
- Some hardware accelerators like DPAA (Data Path Acceleration Architecture) for fast Ethernet processing.

They differentiates by

- The implemented cores and the cache repartition (see Table 16);
- The detailed characteristics of implemented functional blocks and IO drivers – for instance the P4 implement DDR2/DDR3 compatible drivers and the P5 DDR3 compatible drivers;
- The number of IP implemented.

For our purpose, these families can be considered as similar. They have been initially designed for processing in network area applications- see for instance the P4080PB [57].

The following sections describe the main aspects of their architecture, focusing on the P4 and P5 series and more particularly on P4080.

The basic documents available for a microcontroller are listed in the Table 17[49].

Document name	Document description	Examples
Product Brief	Provides an overview of the microcontroller features as well as application use cases.	P4080 Communications Processor Product Brief [57]
Datasheet	Includes all the hardware design related information as power supply specification timings, thermal environment.	P4080/P4081 QorIQ™ Integrated Processor Hardware Specifications [58]
Reference Manual	Describes the features and operation of the microcontroller	P4080 QorIQ™ Integrated Multicore Communication Processor Family Reference Manual [33]
Core Reference Manual	Describes the features of the core	e500MC Core Reference Manual [59]
Programmer Reference Manual (*)	Provides software and hardware designers with the ability to design and program to the instruction set architecture (ISA) defined for embedded environment processors and by Freescale's implementation standards (EIS).	EREF 2.0: A Programmer's Reference Manual for Freescale Power Architecture® Processors [60]
Errata list	Details all known silicon errata on the component.	P4080 Chip Errata [61][50]
Application notes	Many application notes are emitted by the manufacturer in order to configure or use correctly the device. Often these application notes complement and particularise the reference manual and eliminate some ambiguities.	AN3532: Error Correction and Error Handling on PowerQUICC™ III Processors [62]

Table 17: Multicore microcontroller list of reference documents

[49] The document listed here can be found on Freescale site: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=P5020&fpsp=1&tab=Documentation_Tab.

[50] Reference [62] is not publically available; the reader who wishes to glance at an errata list can refer, for instance, to P5020 chip errata [67].

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Table 17 list only public documents useful for the present study. Documents signalled with an (*) are not directly applicable but due to their operational nature they appear to be pedagogical and useful as a complementary source of information.

As already mentioned for single core microcontrollers, some other sources are available

- White papers [63], Publication and patents [55] emitted by the manufacturer have not the official character of a Reference Manual but can help to have an overview on a particular topic and help to ask relevant questions to the COTS manufacturer;
- Due to the complexity of the microprocessor and of its documentation (several thousands of pages) some training are proposed by the manufacturer and by affiliate consultants (for instance the AC6 trainings for Freescale microcontrollers). These training cannot be considered as elements of proof but they can help a lot in the overall understanding of the COTS functioning.

8.8.2 Architecture description

The Reference Manual [33] of the P4080 proposes a block diagram oriented toward the features ensured by the microcontroller. These features are:

- Computation: 8 e500mc cores, each with private 2x32kB L1 cache and 128kB L2 cache,
- Two L3 caches
- Memory features
  - Two 64 bits DDR2/DDR3 memory controllers,
- High speed I/O :
  - Ethernet interfaces,
  - PCI Express 2.0 controllers/ports
  - serial RapidIO controllers/ports
- Additional peripheral interfaces
  - USB 2.0 controllers
  - SD/MMC controller (eSDHC)
  - SPI controller
  - I^{2}C controllers
  - Dual DUARTs
  - GPIO
- Enhanced local bus controller (eLBC)
- Material accelerators feature
  - Two 4-channel DMA engines
  - Data Path Accelerator
- Real Time Debug features
- Multicore programmable interrupt controller (MPIC)
- Power Management

Some of these features are exclusives as they consume the same SerDes lanes.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

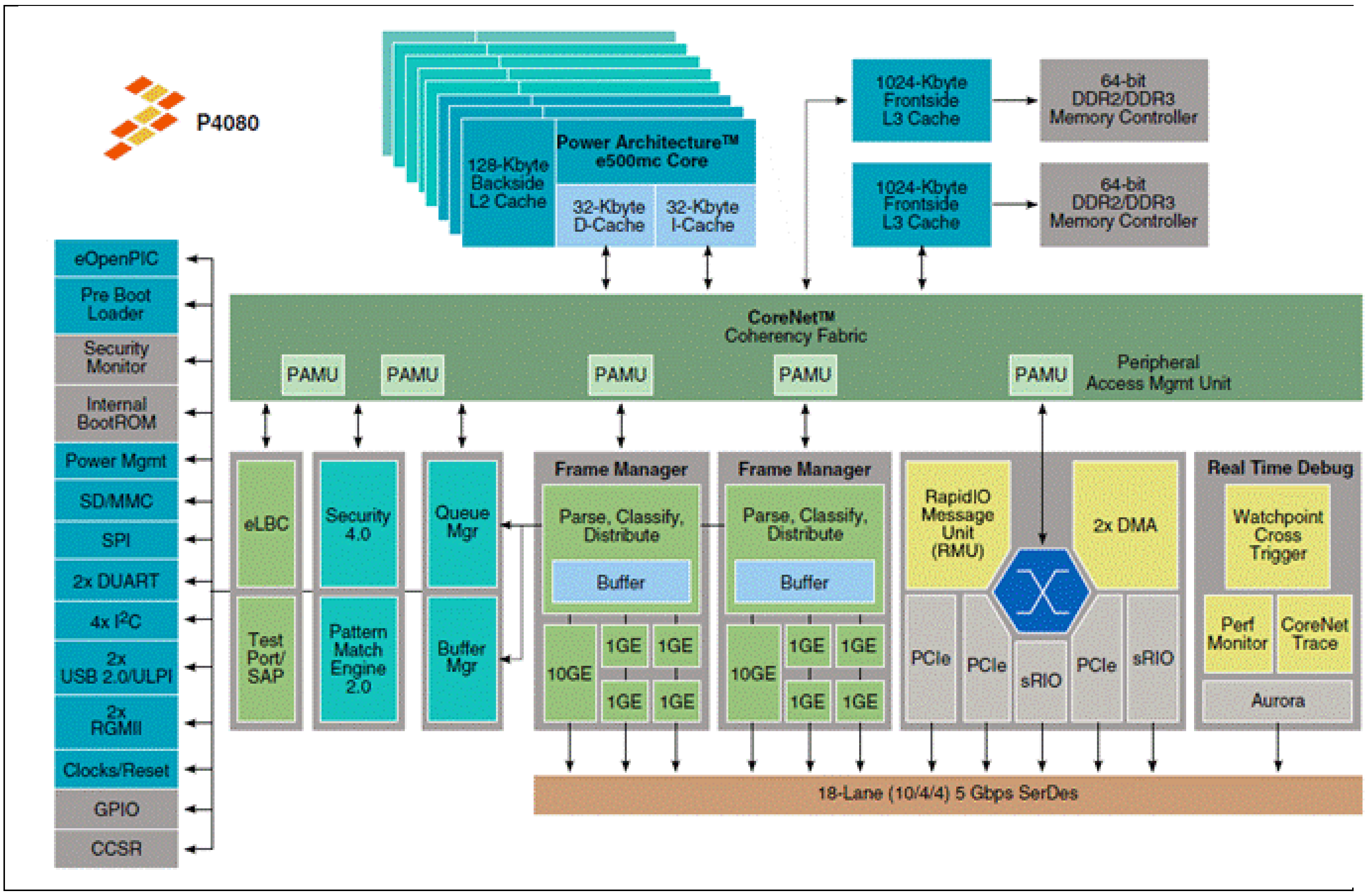


Figure 88 : P4080 block diagram from P4080RM [33].

By many aspects the diagram of Figure 88 appears to be very high level with some approximation and missing links. It is not always simple to determine if some of the blocks identified here are localized in a drawn high level block or spread amongst various other blocks. That's in particular the case with blocks that offers internal services like Power Management or CCSR (Configuration, Control, and Status Register). In such a case some non-mentioned links may exist between these blocks and others.

The next section describes these blocks, their failure modes and the embedded failure mitigation mechanisms.

Note on memory structure of P4080
P4080 microcontroller internally addressed memory mapped as follow:

- Logical, virtual, and physical (real) address spaces within the Power Architecture core(s);
- Internal local address space that includes;
  - Internal Configuration, Control, and Status Register (CCSR) address space;
  - Internal Debug Control and Status Register (DCSR) address space;
- External memory, I/O, and configuration address spaces of the serial RapidIO link;
- External memory, I/O, and configuration address spaces of the PCI Express links.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

By the fact, almost all the blocks are linked to the CCSR or DCSR registers directly or through the Interconnect[51]. These links are not recalled in the next section.

8.8.3 Block study

The following sub-section detail the analysis of the most important blocks depicted on Figure 88.

8.8.3.1 Cores

8.8.3.1.1 Description

P4080 cores are described in the core reference manual (see [59]) and their integration on the chip is described in chapter 7 of the reference manual (see the P4080RM [33]). Each core has independent,

- L1 cache for instructions (32kB) and L1 cache for data (32kB);
- Mixed data and instructions L2 cache (128kB).

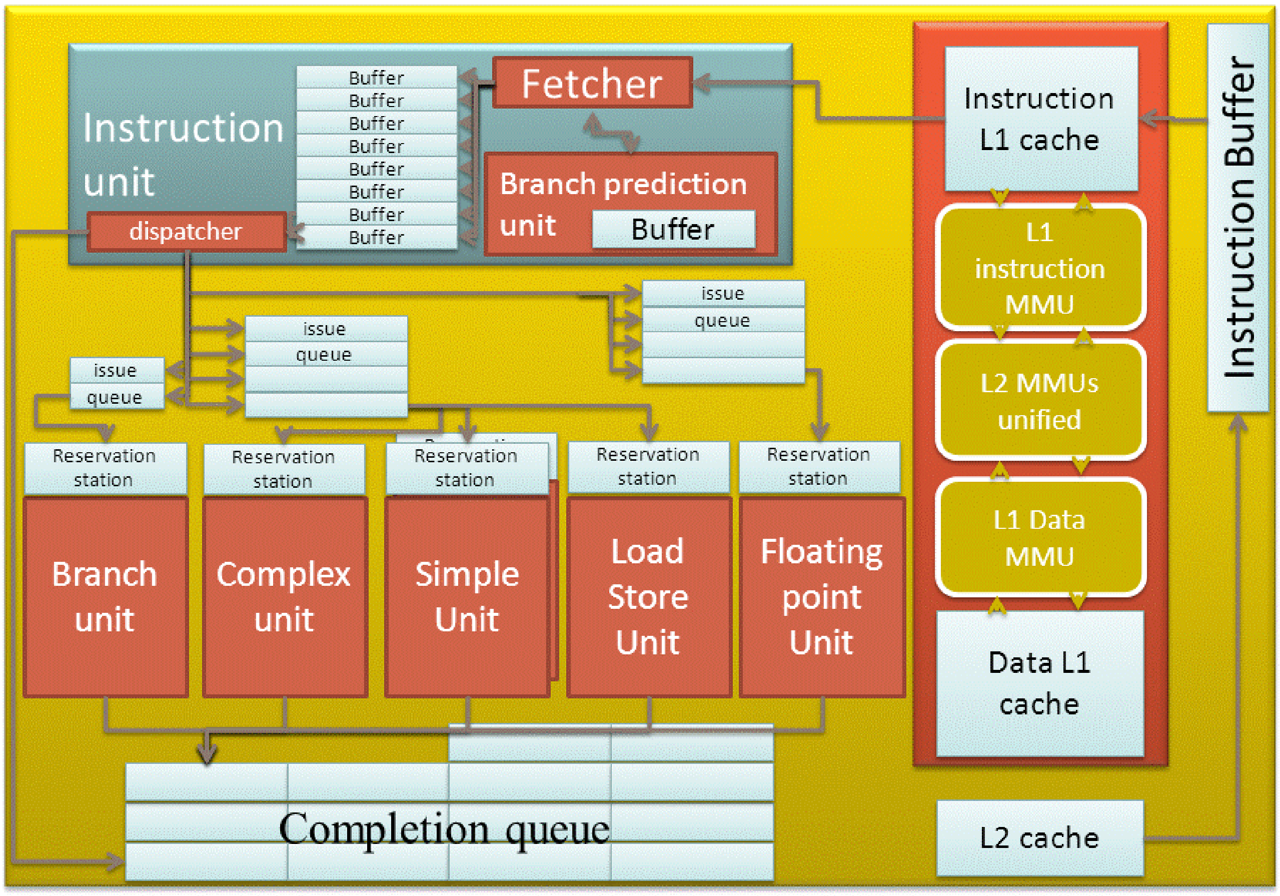


Figure 89: Simplified e500MC block diagram focusing on instruction paths and memory type areas (clear grey zones) – adapted from [59]

E500mc is a 32 bits core[52] implementing several levels of pipelining and different levels of buffer allowing fetching instructions, facilitating branch prediction, queuing instructions, etc.

[51] In the subsequent chapter, when possible, CoreNet™ will be named by its generic name "Interconnect" even if it appears that Freescale CoreNet™ covers more than the simple function of interconnecting cores and I/Os (see 0).

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

Its architecture is very similar to the one of e600 core described on Figure 83

Each e500mc is interfaced with interconnect through its L1 cache (L2 cache is backside).

8.8.3.1.2 Failure modes

At the breakdown and abstractions level tackled here the difference between e600 and e500mc have no impact on failure modes.

As already mentioned in paragraph 8.7.3.1.2, failure modes of the cores are those already listed in the section 6.3 for the interface between hardware and software:

- No program instruction outing;
- Erroneous calculation outing;
- Latency in program instruction outing (Maximum Execution Time drift);
- Inversion of tasks.

Toward its interface with interconnect; each core can develop the standard failure modes developed in the first part of section 6.3. In the particular case of transfer between cores and interconnect the wording "message" as to be considered in a large meaning (burst should be more adapted).

Failures modes of e500mc core are considered similar to those of e600 core described in paragraph 8.7.3.1.2.

8.8.3.1.3 Failure mitigation mechanisms

8.8.3.1.3.1 Parity Checking

A parity checking is configured on L1 Cache for instruction and Data and tags of L2 Cache in order detect an odd number of bit flips. No Error Correcting Codes have been implemented on these caches or information in cache because their allocated because their small volume induces a low probability of bit considering soft errors, namely low energy alpha particle (from the package), high energy thermal particles and thermal neutrons. This rationale induced that this parity check (as well as the Error Correcting Code outlined in the next paragraph), is not designed to cover design errors. Tests described in chapter 9 verifies that this assumption is respected.
Note that in order to be operative, this parity check has to be activated in core registers as described in core reference manual [59] and outlined in Freescale Application Note AN3532 [62]:

- For L1 data, in L1CSR0 (L1 Cache Control and Status Register 0)
- For L1 instruction, in L1CSR1;
- For L2 tags, in the L2 Cache Configuration Register L2CFG0.

In case of error detection by the parity check, the relevant sanction is configurable (in the already cited registers) and applied directly by the core without reference to the MPIC except for possible error reporting.

[52] E500mc core is an extension of e500 cores for multicores (mc). The 64 bits extension appeared with e5500 (P5 series) and the multithreading with e6500 (T series).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.8.3.1.3.2 ECC (Error Correcting Code)

An Error Correcting Code (ECC) which encoding is described in the Freescale Application Note AN3532 [62] is configured on processor memories. It correct one bit errors and detect systematically 2 bits errors. As already noted for the parity check, this ECC is not designed to cover design errors but soft errors. It is applied at core level on L2 data and is configurable in the L2 Cache Configuration Register L2CFG0.
The implemented ECC mechanism offer a self-test mechanism through error injection in the L2 (see core reference manual [59] section 2.15.4.9 and EREF manual [60]). During this test phase an error is inserted in the L2 cache and the ECC status is checked to be coherently activated in the L2 Cache Error Capture ECC Syndrome Register.

Note: Other memories units included in the core are not covered by any parity or ECC mechanisms. This the case for instance for

- Buffers outlined in Figure 89
- MMU internal Translation Lookaside Buffers (TLB),
- Various core related configuration and status registers.

Following argument can be given for such choice:

- The size of each of these memories is considered as negligible compared to the L1 and L2 sizes (TLB global size is 90 B);
- Due to intensive use of the buffered data even parity should be to time consuming.

Although, Cache protection by parity or ECC is now in the state of the art of microprocessors (see for instance ARM Cortex-R series processors), buffer protection is still a research topic which merits are is debated.

8.8.3.1.3.3 MMU (Memory Management Unit) [63]

Each core embeds a MMU that controls all the address-based accesses initiated by the cores.
Its primary purpose is to map the Effective Addresses manipulated by application software to the Real Addresses of the local SoC mapping.

In addition to this function MMU is a protection barrier that filters each memory-mapped access in order to prevent forbidden access of the core to a memory page. This filtering is performed through an entry in a Translation Lookaside Buffer (TLB) and depends on the type of access and privilege level for read, write or execute.

With this function, MMU support spatial partitioning.

Note: MMU cannot be deactivated by configuration.

8.8.3.1.3.4 Embedded Hypervisor

For a general review, please refer to [63].

In the case of Freescale multicore series (QorIQ^{TM} P3 and above), processors embed hardware assist to ease the implementation of a virtualization layer, usually named hypervisor. Each core supports three levels of instruction privileges: user, guest supervisor (used for guest Operating Systems), and hypervisor (also called host supervisor). The piece of software executed in hypervisor mode is granted exclusive rights to

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

configure the Memory Management Unit (MMU) and some core's Special Purpose Registers (SPR) as specified in the corresponding reference manuals [59]. Guest software attempts to reconfigure the MMU and SPR trigger a Privilege Exception that is handled by the hypervisor.

Therefore an embedded hypervisor has the ability to control unobtrusively operating systems' use of hardware resources such as the main memory and peripherals (PCIe, UART). Furthermore, by trapping any attempt to access directly a reserved resource, an embedded hypervisor can proxy this access on behalf of the original requestor within predefined specifications. One can refer to resource virtualization techniques for more information [64]. That prevents malicious or faulty software from invalidating platform's dependability properties. That usage is relevant in open and/or multi Operating Systems environments as it makes dependability requirements enforced through a single piece of software.

The market of embedded hypervisors contains both commercial solutions (VxWorks, Integrity, PikeOS, Xtratum, Enea, etc.) and open solutions (Xen, KVM, Topaz, etc.). Most of those solutions were derived from real-time operating systems and/or microkernels (e.g. L4 family) that were designed for safety critical embedded applications. Home-maid hypervisors can also be developed for specific devices and applications, such as IMA systems [65].

8.8.3.2 Interconnect: CoreNet^{TM} Coherency Fabric (CCF)

8.8.3.2.1 Description

Freescale CoreNet^{TM} Coherency Fabric (CCF) is described in the Chapter 9 of P4080RM [33]. Its basic function is to ensure multiple parallel transactions with retry facilities, low latency and high bandwidth.

Freescale CCF acts as a central interconnect for cores, platform-level caches, memory subsystems, peripheral devices, and I/O host bridges in the system (see Figure 88). All of them are connected to CoreNet^{TM} either directly, either through PAMU or/and OCeaN^{TM}.

Detailed information on CoreNet^{TM} is difficult to obtain and to validate considering the confidentiality level maintained on this topic. Information on such block should be cross-checked from various public sources: reference Manual [33], academic papers and Freescale patent [66].

From these sources, CoreNet^{TM} appears to be "interconnect" of the type switched network mentioned in section 5.3.4.2 and represented on Figure 90.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

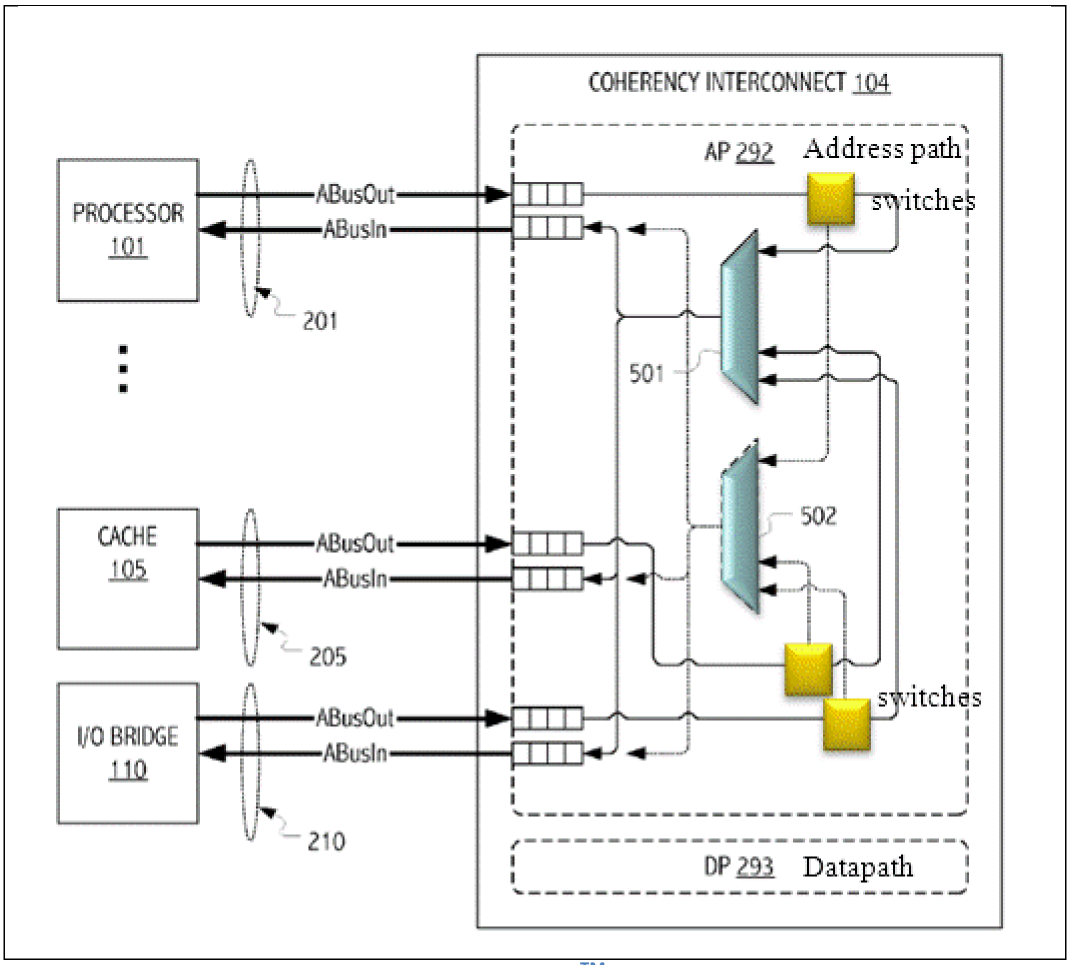


Figure 90: A possible view of an interconnect similar to CoreNet TM with three interfaces. This figure is derived from [66].

In addition to this connection function, Freescale CoreNet TM has also a function of hardware acceleration for some operation. It is thus possible to transfer to CoreNet TM a decorated instruction which decoration is treated directly by CoreNet TM. Such an instruction can be generated by a Core or by the PAMU that can be configured in order to decorate instructions acting for instance on some memory zone. A typical example of decoration is the counting of data copied by the DMA from PCIe to the DDR memory. This allow core to be discharged from any action in such operation.

CoreNet TM configuration is defined and implemented by Freescale in some reserved zone of the configuration register.

8.8.3.2.2 Failure modes

On each of its interface interconnect can develop the different usual failure modes. Figure 90 can be used as a guide in order to list the possible failures of this block.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

8.8.3.2.2.1 Loss of Messages

Even if the Reference Manual claims that no transaction can be lost by interconnect (§ 9.1.1), it seems important to consider this possibility. It can be noted that such a type of error has been listed in errata of P5020 (A-004510 in [67]) before to be solved in version 2.0 of the chip.

Loss of messages can take the two following forms:

- Loss of data type message:
  In addition to direct loss of message, it can be loss because addresses are corrupted or lost
- Loss of instruction type message:
  Interconnect transfer programme instruction from Flash memory to DDR3 during the boot and between DDR3 memories and caches during normal using phase. It is possible to loss this instruction during this transfer.

8.8.3.2.2.2 Untimely transfer of message

From Reference Manual P4080RM [33], it is known that CCF implement a retry process in order to avoid message lost. This mechanism could lead to babbling through a mechanism similar (although certainly not identical because the protocol and the involved technologies are different) to the one described on PCIe interface in subsection 7.7.2.2.
As mentioned on Figure 90 and confirm by the reference manual, interconnect contains buffers that could store transaction during few clock pulses in order to wait that a way is free. The presence of such buffer could also generate untimely resending of messages in case of pointer error.

8.8.3.2.2.3 Abnormal sequence of messages

The two afore mentioned mechanisms, which could lead to untimely transfer of information, could also lead to possible abnormal sequence. In particular a loss of message compensated by a retry could make the corresponding message arriving after a message sent before it.

In another way, a buffer pointer error could invert two messages.

8.8.3.2.2.4 Untimely or forbidden transition of information

An information (address, data, instruction) stored in a buffer could suffer from a bit flip and then to an untimely transition. An ECC mechanism is settled on CoreNet TM in order to mitigate effects of this behaviour.

8.8.3.2.2.5 Impossible transition of information

The bit flip phenomenon described in preceding paragraph could equally be a bit stuck phenomenon and then lead to an impossible transition.

8.8.3.2.3 Failure mitigation mechanisms

CoreNet TM implements various failure mitigation mechanisms that can cover the failures listed in the previous paragraph:

- Coherency violation detection when a state of the coherency granule was found to be in violation of the coherency protocol prevent from untimely and impossible transitions of addresses.
- Local Access Error detection that prevent from untimely and impossible transitions on addresses:

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

  - Local Access Window Miss. An incoming transaction misses all LAWs.
  - Unavailable target ID programmed in LAW attribute register.
- Retry mechanisms that prevent for the loss of transaction;
- Transaction ordering support that prevent from abnormal sequence;
- ECC on buffers that prevents from untimely and impossible transitions.

8.8.3.3 Peripheral Access Management Unit (PAMU)

8.8.3.3.1 Description

PAMU is described in chapter 10 of P4080RM [33]
The PAMUs reside at the interface between what is considered as a coherency zone (interconnect, cores and DDR drivers) and the IO domains (see Figure 88).
Complex peripherals (included DMA) can interact with the CoreNet TM and intrude to memory. PAMU plays an equivalent role as those played by MMU and enforces authorization and access control into the coherency domain. More generally PAMU has three main roles:

- Check access rights of an I/O or a DMA to some physical addresses;
- Address translation from logical I/O addressing to memory physical addressing;
- Instructions translation from peripheral IP protocol for operations to CoreNet TM protocol, for instance it can forbid to an I/O to perform decorated reading / writing.

The last functionality is linked with the previously mentioned functionality of CoreNet TM related to decorated instructions. (See paragraph 8.8.3.2.1)

In order to perform this access right checking, the address maps necessary to PAMU are stored in DDR in Peripheral Access Authorization and Control Tables (PAACT). When a master I/O try to access the coherency domain through PAMU (see Figure 91), PAMU check its Logical I/O Descriptor Number (LIODN) with entry (PAACE) of PAACT stored in PAMU Cache (1). In case of cache miss, PAMU fetches corresponding PAACE value from DDR to cache (2). If no access violation is detected the corresponding transaction is acquitted. If not an access violation status is raised.

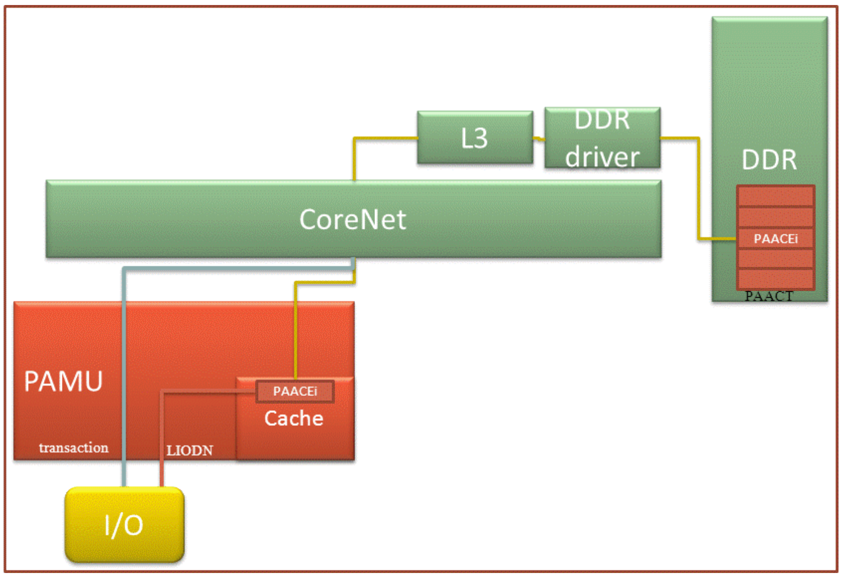


Figure 91: PAMU access right checking simplified process.

Open document analysis allows an indicative mapping of PAMU instances to different peripheral:

- PAMU1:
  - Local Bus (eLBC) and other I/O and internal IP (see P4080RM [33])
  - Part of DPAA concerning Security Manager, Pattern Match Engine and Rapid IO Manager;
- PAMU2:
  - Part of DPAA concerning Queue Manager, Buffer Manager and the RAID (Redundant Array of Independent/Inexpensive Disks);

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- PAMU3: DPAA: Frame Manager;
- PAMU4: OCeaN™ Switch Fabric and thus PCIe, SRIO (Serial Rapid IO) and DMA.

Although this mapping is coherent with Figure 88, the following sentence of [33], let suppose that there be only one PAMU with 16 entries:
"The PAMU is partitioned into 16 identical instances. Not all are necessarily backed with physical hardware. However, all of them must be programmed identically or undefined behaviour may result." [33] p. 121.

8.8.3.3.2 Failure modes

PAMU is a protection block. It should protect against abnormal behaviours of the peripherals or material accelerators in interface. In case of failure, the embedded protection mechanisms implemented in PAMU can impact transactions sent on the CoreNet ™.

8.8.3.3.2.1 Loss of Messages

Due to its protection function, PAMU can untimely block some transaction considering that they are not addressed to the right zone. This can be generated by a corruption of the PAACE (Peripheral Access Authorization and Control Entry) within PAMU Cache or in DDR.

8.8.3.3.2.2 Untimely transfer of message

There is no evidence that PAMU implement data cache or buffers so this behaviour is not considered here.

8.8.3.3.2.3 Untimely or forbidden transition of messages

When activated, the translating addresses mechanism implemented by PAMU can corrupt address (untimely or forbidden transition on addresses or impossible transition on addresses). This phenomenon will erase and replace a data in the memory zone dedicated to the considered I/O (Figure 92 - (6)) unless a second error occurs in the LIODN check. In this last case a breaking of spatial partitioning may occurs.

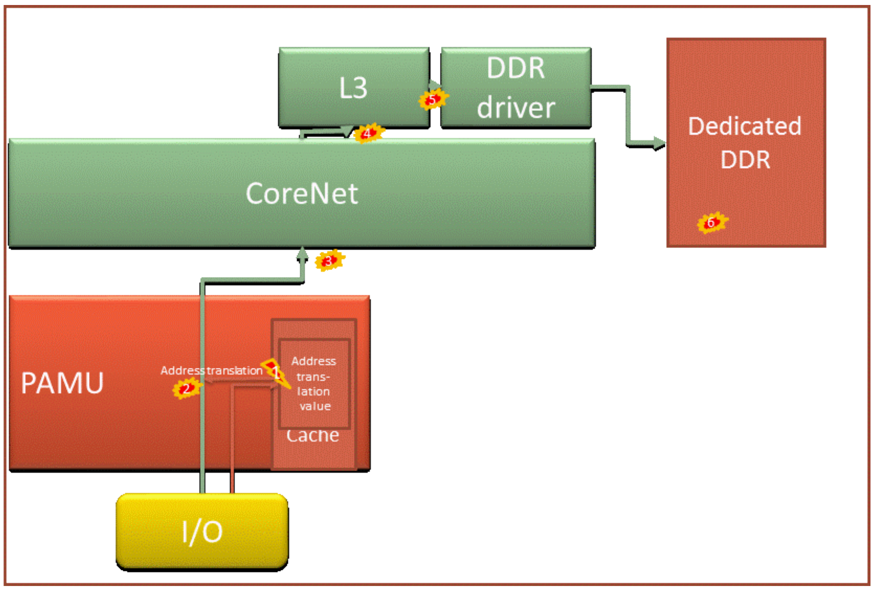


Figure 92: Address translation fault mechanism.

These failures can be caused by corruption of some table entries

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

involved in address translation mechanisms, in cache or in DDR.

8.8.3.3.2.4 Impossible transition of information

See previous paragraph.

8.8.3.3.2.5 Abnormal sequence of messages

There is no evidence that PAMU implement data cache or buffers so this behaviour is not considered here.

8.8.3.3.3 Failure mitigation mechanisms

As outlined in description section, main function of PAMU is dedicated to failure mitigation mechanism. Indeed PAMU help ensuring spatial partitioning (note that it has no influence on some time partitioning). The failure mechanisms described in previous paragraph may be caused mainly by additional PAMU features. All involved corruption of some tables in DDR or in PAMU cache.

It has to be noted that the ECC mechanism described in paragraph 8.8.3.1.3 about core L2 caches is also implemented on PAMU cache so that single bit flip will be corrected and double bit flip detected.

8.8.3.4 CoreNet ᵀᴹ Platform Cache (CPC)

8.8.3.4.1 Description

CoreNet TM Platform Cache is described in Chapter 8 or P4080RM [33]
The CoreNet TM platform cache (CPC) is a CoreNet TM -compliant target device that functions as a general purpose write-back cache, I/O stash and memory mapped SRAM device, or a combination of these functions.

- As a general purpose cache, it manages allocations and victimizations to improve read latency and bandwidth over accesses to backing store (for example, DRAM). As an I/O stash, it can accept and allocate writes from an I/O device in order to reduce latency and improve bandwidth for future read operations to the same address.
- As an SRAM device, it acts as a low-latency, high-bandwidth memory that occupies a programmable address range.

The two CPC are interfaced with the CoreNet TM on one side and one DDR3 memory controller in an inline configuration.

8.8.3.4.2 Failure modes

As memory CPC can develop the common following failure modes. Moreover the CPC is highly configurable and corruption of this configuration in CCSRBAR may strongly change the behaviour of CPC.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.8.3.4.2.1 Loss of messages

Loss of messages is a basic mode of memory - although SRAM is reliable. Some configuration of CPC could make loss of data if corrupted.

8.8.3.4.2.2 Untimely transfer of messages

This transfer is not initiated by the CPC itself. This mode has to be imputed to agents that can act on CPC:
- CoreNet TM,
- Cores (through CoreNet TM and PAMU)
- Peripheral (through CoreNet TM and PAMU) and in particular DMA
- Etc.

8.8.3.4.2.3 Untimely or forbidden transition of information

A bit flip can change the information before it is read.

8.8.3.4.2.4 Impossible transition of information

A bit stuck can prevent information to be changed when it is written.

8.8.3.4.2.5 Abnormal sequence of messages

Inversion in the delivery order of messages cannot be excluded although it depends upon design details under NDA information.

8.8.3.4.3 Failure mitigation mechanisms

CPC includes separate ECC mechanisms on tag and data
CPC includes error injection mechanisms.
Errors detected are reported in a register.

8.8.3.5 Multicore Programmable Interrupt Controller

8.8.3.5.1 Description

The MPIC (Multicore Programmable Interrupt Controller) is a centralized resource that
- Concentrates internal and external requests for processes interruption;
- Prioritizes these interruptions;
- Applies them and
- Manages the interruption follow-up.

The interruption sources are mainly:
- External events;
- Internal SoC events (112 sources for P4080);
- MPIC events based on
  - Inter-processor interrupt channels,
  - Message Signal Interrupt from PCIe,
  - Message interrupt channels from MPIC,
  - MPIC global timers.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Note that MPIC do not managed core driven interruption outlined, for instance, in the case of Cache Parity or ECC errors. These interruptions are directly managed by the core (see Figure 93).

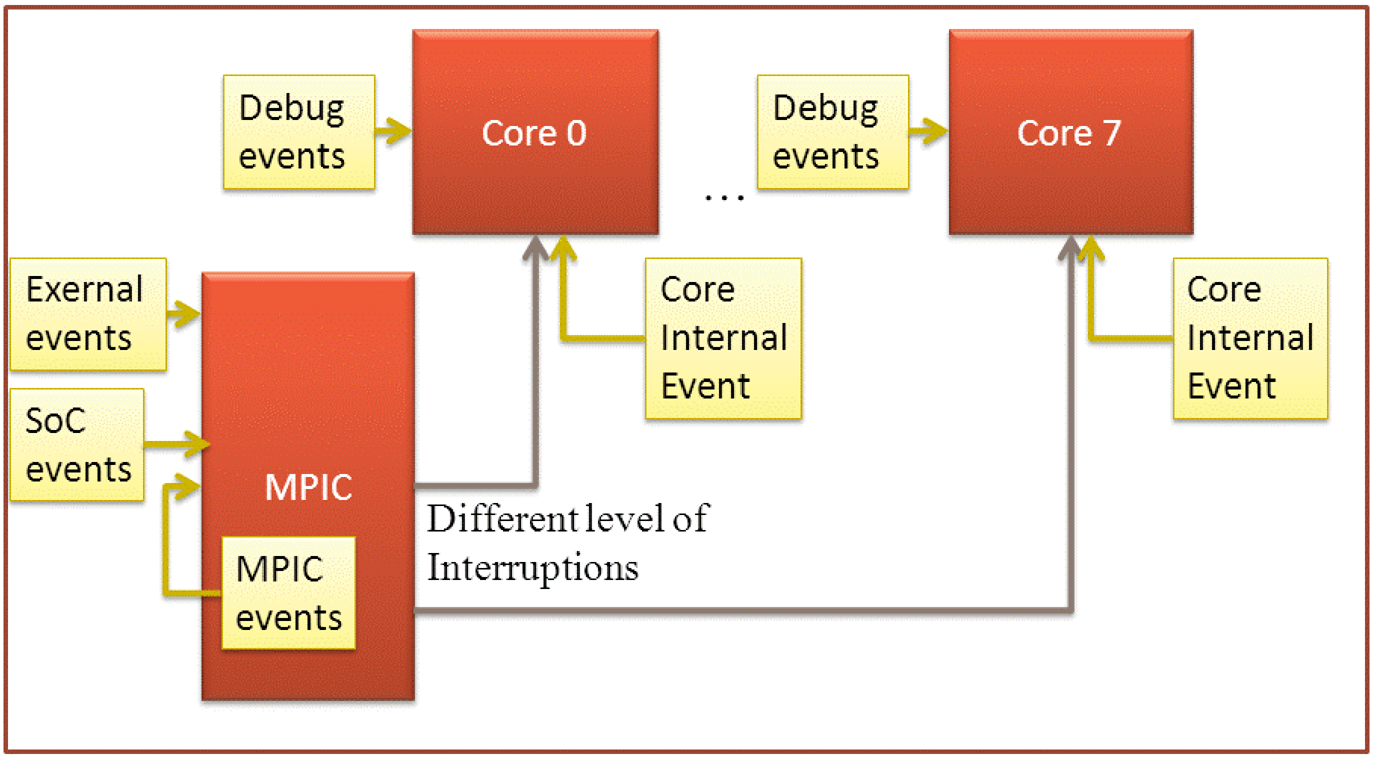


Figure 93: Simplified view of interruption mechanisms (see [33] and [63]).

8.8.3.5.2 Failure modes

8.8.3.5.2.1 Loss of message

The loss of some critical IT will make impossible to raise a diagnostic on the multicore and then to reset it.

8.8.3.5.2.2 Untimely transfer of message

Untimely raising of non-critical IT may temporarily interrupt the running applicative partition and slow down the execution creating Maximum Execution Time drift and potentially erroneous execution if the partition execution exceeds its allocated time slice.

Untimely (advanced) raising of critical IT when no rising is requested could lead to a global reset of the COTS.

Untimely (late) raising of critical IT could delay a reset and lead to erroneous execution of the partition.

8.8.3.5.2.3 Abnormal sequence of messages

No scenario found in the abnormal sequence of two interruptions.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.8.3.5.2.4 Untimely or forbidden transition of information

Untimely transition of an interruption from one nature to another could lead to wrong diagnostic of the core to be reset (in case of partial reset) or to other erroneous decision.

8.8.3.5.2.5 Impossible transition of information

See Untimely transfer of message

8.8.3.5.3 Failure mitigation mechanisms

Despite of their role on detection and/or mitigation of microcontroller errors (see Mixed mechanisms on section 9.3.3), no internal mitigations have been found that could control the PIC. Section 9.3.3 implements mechanisms that use the MPIC.

8.8.3.6 CCSR (Configuration, Control, and Status Registers)

8.8.3.6.1 Description

In such a configurable object as a multicore microcontroller is configuration register is a crucial point.
In paragraph 8.2.3.3 a small preview of the possibilities and associated risks offered by these configuration registers have been explored. In a microcontroller like the P5020, the size of these registers is 16 MB and is comparable on P4080.
It groups in one hand activations and fine tuning variables of each block, IP or features of the microcontroller and in the other hand status of these blocks, IP or features.

Table 28 of Annex 1 shows the range of addresses for each blocks, IP or features of the P5020. It is similar for the P4080 even less documented.
This table calls for several remarks the block and features covered and the relative volume of reserve ranges compare to accessible range.
Indeed, it is important to note that CCSR contains:

- Customer accessible configuration,
- Manufacturer reserved configuration,
- Open and private status.

From Table 28 it can be computed that at block level 90% of the 16 MB are reserved by the COTS manufacturer. These zones can be reserved for hidden configuration purposes but at this level, more certainly, for future development purposes.
At block configuration level zones are again reserved or not documented. For instance considering CoreNet ^{TM} Coherency Fabric CCSR zone, ranging from 0x01_8000 to 0x01_8FFF, addresses from 0x01-8000 to 0x01-8680 (1664 B) and addresses from 0x01-8A18 and 0x01_8FFF (1511 B) are not documented and could contain hidden configurations or status.

This pattern, the criticality of configuration variables and the intellectual property stakes defended by the COTS manufacturer, lead to some possible strategies for the protection of CCSR. This protection is discussed in next chapter.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

8.8.3.6.2 Failure modes

8.8.3.6.2.1 Loss of message

As already noticed for bridges, a configuration register can be corrupted but not lost.

8.8.3.6.2.2 Untimely transfer of message

In addition to being accessible by the processor cores, the CCSR are accessible from external interfaces in particular SRIO and PCIe. This allows external masters on the I/O ports to configure the device (P4080RM [33]).

- An untimely access of PCIe (for instance) to CCSR could untimely change a part of the configuration.

8.8.3.6.2.3 Abnormal sequence of messages

8.8.3.6.2.4 Untimely or forbidden transition of information

First entries of Table 28 “Local access control-Local configuration control” are in fact the configuration of the localization of the configuration table.

- Such auto reference could lead to a global untimely transition of the value of the configuration table to a valid albeit incorrect value or to a forbidden transition to a value that does not correspond to an admissible configuration (in totality or partially). In any case, the behaviour of the microcontroller would be in this case unpredictable.

External masters do not need to know the location of the CCSR memory in the local address map. Rather, they access this region of the local memory map through a window defined by a register in the interface programming model that is accessible to the external master from its external memory map.
In particular, the PCI Express controller's base address for accessing the local CCSR memory is selectable through the PCI Express CCSR base address register (PEXCSRBAR), at some offset (P4080RM [33]).

- A normal access for configuration of PCIe with a corrupted PEXCSRBAR could lead to a shift in the zone written through this configuration transaction and by the way to a major untimely or forbidden transition of the configuration. In this case the behaviour of the device could be changed depending on the zone modified.

- More basically and locally, an untimely modification of a configuration bit could for instance activate a deactivated block or modify the behaviour of a block. This modification could be easily detectable or not.

8.8.3.6.2.5 Impossible transition of information

Applied to configuration register, this mode is “impossible transition of configuration”. It is only relevant in case of configuration change request through PCIe. In this case a PCIe initiated change of configuration should not be effective.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

8.8.3.6.3 Failure mitigation mechanisms

There are no particular protections on CCSR. Some architectural detection and mitigation means are dedicated to configuration registers (see section 9.3.4.2).

8.8.4 Concluding remarks

Many aspects of microcontroller concluding remarks are applicable here. The prominent aspects of multicores compared to single core are the generalization of interconnect crossbar in their design and the presence of drastically more transaction initiators that can generate abnormal behaviour (fir instance untimely transfer of message and even loss of message) through contention phenomenon on interconnect. This interconnect implement itself some mechanisms in order to avoid or to mitigate this phenomenon. Modern microprocessors increase their interconnectivity and in particular the possibility to interact directly with the memory through I/O like PCIe. In order to control such operation multicores (at least those embedding interconnect) implement mechanisms like IOMMUs (PAMU in Freescale context) that prevent from untimely or forbidden transition of address accessible through I/O or DMA. These mechanisms are synthetize in section 9.3.2 and complemented by mixed and by architectural mechanisms in the following sections.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9 DETECTION ISOLATION AND MITIGATION OF COTS DESIGN FAILURES

9.1 INTRODUCTION

Previous chapters list, for various different COTS through their internal structure or their interfaces, a large panel of potentialities. This study mapped a failure meta-model on each element analysis in order to discover the potential failure it could develop. The study tried to stay as open minded as possible, keeping open failure modes even if no credible scenario were found.

The present chapter aims elaborating a global strategy for detection, isolation and mitigation of the failures discovered previously. It appears by the way that COTS internal failures, caused by design errors, described in the previous chapters might be detected isolated and their effect mitigated at various level.

These mechanisms are organized and described in next subchapters in a two main steps process.

The first step of the method is the formalization of a specification of the needs the user has for the COTS and the corresponding tests. These tests covers domain like functional, endurance tests, worst case tests, limit tests,. Even if some of these tests seem to be more functional tests than error detection, isolation tests, we show how to exploit them toward this purpose.

Tests mark off a confidence perimeter for usage of this COTS for which requested performances are ensured and within which no failure modes due to potential design errors can occur.

The bounding is realized in this steps by COTS configuration and when possible by implementation of usage limitation by configuration and guest OS.

The second step is the definition of mitigation means in order to guarantee that the usage of this COTS stays in this perimeter all along its life.

Detection and mitigation mechanisms can be internal to the COTS, mixed or purely architectural (see subchapter 9.3).

The detection and mitigation mechanisms tested either at COTS internal level and at architectural level should be tested through fault injection (see subsection 9.2.3.4).

COTS-AEH_Suggestion_1.	<u>Use of internal detection / mitigation mechanisms</u> In global mitigation strategy definition, the internal mechanisms selected should be: - Specified; - Managed: activation mode, configuration, error handling; - Tested.

<u>Note</u>: Application of the suggestions contained in this report cannot be requested. Nevertheless if a suggestion is applied, the activities proposed should be realized.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

Figure 94 presents this general procedure.

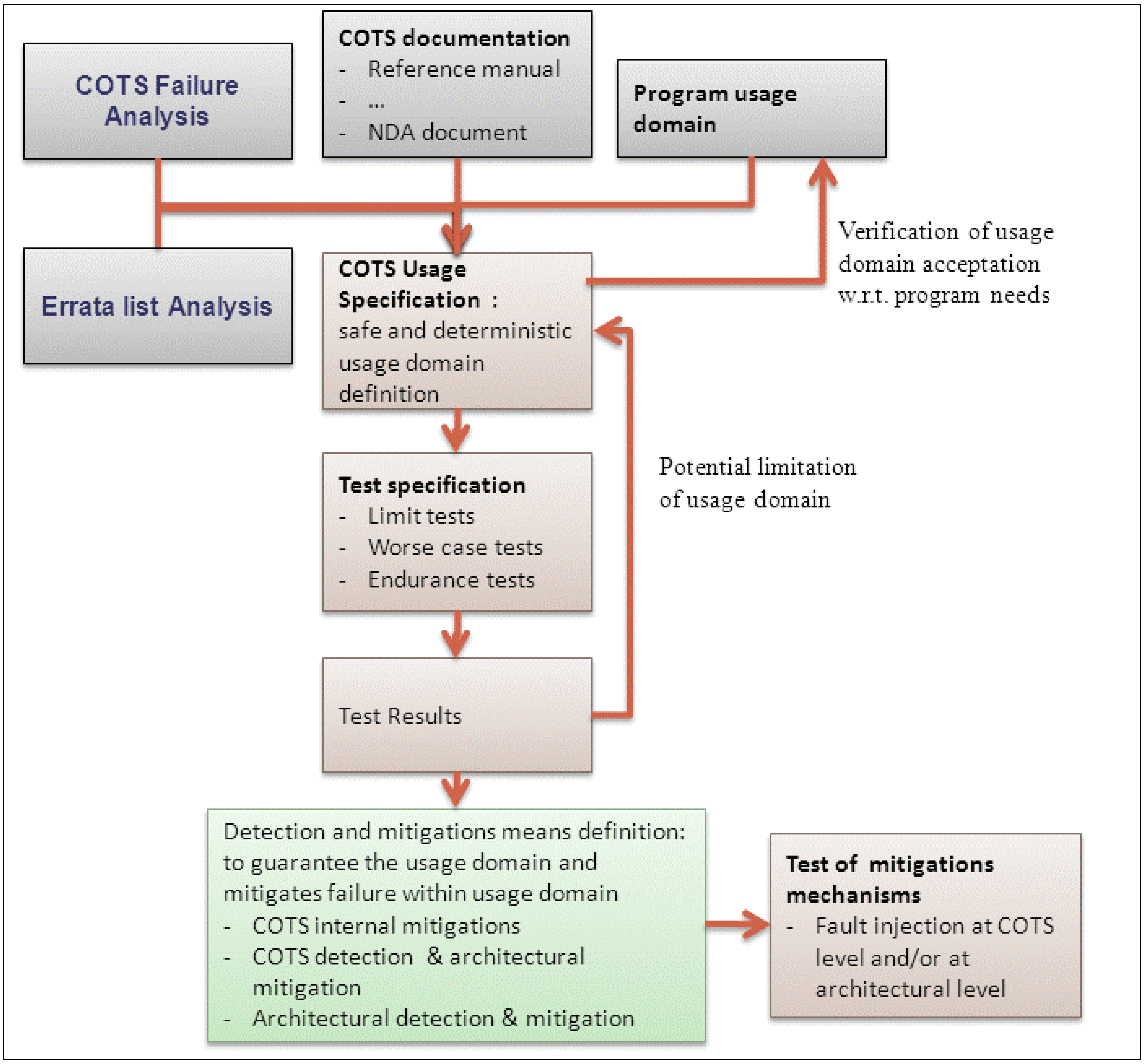

COTS Failure Analysis
COTS documentation
- Reference manual
- …
- NDA document
Program usage domain
Errata list Analysis
COTS Usage Specification : safe and deterministic usage domain definition
Verification of usage domain acceptation w.r.t. program needs
Potential limitation of usage domain
Test specification
- Limit tests
- Worse case tests
- Endurance tests
Test Results
Detection and mitigations means definition: to guarantee the usage domain and mitigates failure within usage domain
- COTS internal mitigations
- COTS detection & architectural mitigation
- Architectural detection & mitigation
Test of mitigations mechanisms
- Fault injection at COTS level and/or at architectural level

Figure 94: General mitigation process.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9.2 COTS SPECIFICATION AND TEST BASE MITIGATION

9.2.1 Inputs data and associated studies

9.2.1.1 Program Usage Domain

First objective of COTS specification is to define the need for the COTS. The first input for this specification is thus the program usage domain for the COTS defined on the basis of general needs (in particular for the microprocessor) and of system or hardware architecture for some others COTS (for instance a bridge).
This need analysis allows to list and to characterize the functions waited for the COTS:

- Computing function: Mega Instruction Per Second (MIPS), Floating point Operations Per Second (FLOPS) etc.;
- I/O type and number needed: PCIe, PCI, SPI, etc.;
- …

Such an approach was already the starting point of [6] and has been normalized in [5]. Recommendations of [6] and requirements of [5] are applicable here even if they are not systematically recalled in order to focus on failures mitigations. For instance the compilation of documentation – public or under NDA, the adequacy between the compiled documents and the SoC version, etc. is prerequisite to the approach proposed here. Points of [5] and [6] rose in this chapter are only those for which we propose refinements.

The first objective of this program need analysis, enriched by system, hardware and software preliminary considerations, are to justify the choice for the COTS to be used.

It will be a major input for COTS specification when COTS will be selected.

9.2.1.2 COTS documentation

COTS documentation should be analyse in order to

- Confirm the adequacy with the need;
- Identify the COTS functions;
- List the logical block and their functions.

These first elements are necessary and sufficient to build a first functional version of the COTS specification (see section 9.2.1.4 for developments).

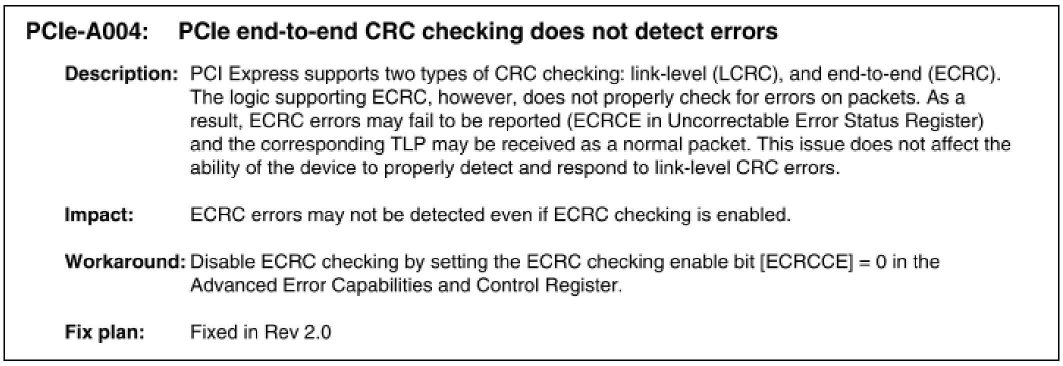
9.2.1.3 Errata exploitation

As soon as COTS and its features are selected, known errata have to be analysed in order determining their applicability and safety impact. Possible workaround have to be evaluated[53] starting from, but not limiting to, those proposed by the COTS manufacturer. It has to be reminded that COTS manufacturer does not know the COTS foreseen usage and that among these usages, airborne applications are very particular. Workaround proposed by the COTS manufacturer may not be applicable in an airborne environment. For

[53] In some cases, the bug will be fixed in a new release of the COTS (see Table 18). If this release is delivered before the beginning of the test, the corresponding erratum has to be kept under supervision but no workaround could be considered.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

instance a CRC that is not correctly decoded cannot be simply deactivated (e.g. Table 18). It has to be replaced at OS level by a corresponding protection.



PCIe-A004: PCIe end-to-end CRC checking does not detect errors

Description:	PCI Express supports two types of CRC checking: link-level (LCRC), and end-to-end (ECRC). The logic supporting ECRC, however, does not properly check for errors on packets. As a result, ECRC errors may fail to be reported (ECRCE in Uncorrectable Error Status Register) and the corresponding TLP may be received as a normal packet. This issue does not affect the ability of the device to properly detect and respond to link-level CRC errors.
Impact:	ECRC errors may not be detected even if ECRC checking is enabled.
Workaround:	Disable ECRC checking by setting the ECRC checking enable bit [ECRCCE] = 0 in the Advanced Error Capabilities and Control Register.
Fix plan:	Fixed in Rev 2.0

Table 18: An erratum example from [67].

9.2.1.4 COTS failure analysis

A COTS failure analysis not oriented to any failure rate breakdown but to list COTS potential failures due to design errors[54] at black box and grey box levels as performed in the preceding chapters enriched the specification and the resulting test plan.

The analysis should be tested back to back with the observed errata. Even if errata result in general from complicated scenarios, they should be expressed as a basic failure mode of the block. The example of Table 18 corresponds to an impossible detection of corruption by a CRC (see subchapter 7.7.). In the previous failure model, it corresponds to “impossible transition of ECRC error status register to error". Errata should thus be classifiable and classified into the category of failure modes discovered in the failure analysis. In case of impossibility, Failure analysis should be enriched in a coherent[55] process of lessons learnt integration.

9.2.2 COTS specifications

The specification can be built on the basis of previous inputs. Typical structures encountered are

- A black box specification describing the main functions of the COTS requested by the program and their characteristics. It has to be organized in requirements and covers
  - The features requested for the COTS
    For instance, computation capabilities, 2 PCIe 4x; 1 SPI, 2 DMAs; etc. ;
  - The COTS features that have to be deactivated;
  - The failures of COTS outputs associated with used features or deactivated features.
- A grey box design document describing the blocks, their features and configurations
  - The blocks necessary in order to realize the requested features;
  - The block that have to be inhibited   nd the way to inhibit them;

[54] This analysis will be later useful for COTS manufacturing errors preview and will be also exploitable for random failure analysis.
[55] The objective, here, is not to add a new found mode (e.g. from errata) to a list of pre-existing failures but to extend the failure model homogeneously and consistently.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

- The configuration of activated blocks in order
  - to ensure the requested features and
  - to prevent against the appearance of unwanted features;
- The COTS internal failure modes identification from Airborne equipment developer with
  - Failure modes observed in errata from COTS manufacturer ;
  - Potential failure modes identified in Analysis of airborne equipment developer;
- The specification of workaround for failure modes that are effectively seen in errata;
- The way to detect and mitigate the failure modes;
- The detection mechanisms guaranteeing in operation that the detection and mitigation mechanisms defined below are operational.

A particular importance is coated by means of deactivation and more generally by configuration as outlined in paragraphs 8.2.3.3 and 8.8.3.6.

9.2.3 Tests with regard to specification and usage domain limitation

Test plan is based on COTS specification and aims at specifying test cases allowing verifying primarily that the COTS realizes correctly its intended functions with waited performances (MIPS, WCET, etc.).

As already outlined, it is not possible to define Point of Control (PoC) or Point of Observation (PoO) inside COTS, so typical test cases will rely on external stimulus and external observation in order to test this correct implementation of COTS features (Figure 95).

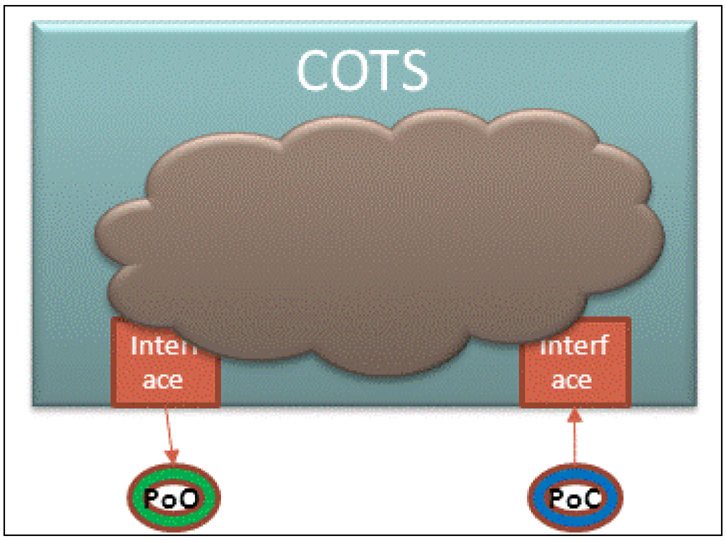


Figure 95: The simplest black box test.

To notable exception may be noted:

- In case of programmable COTS, it can be considered that at some point the test program executed by the core is known and that it induces a PoO;
- The data transfer between a COTS and a memory is not observable (see subsection 5.3.3.1). Memory should be qualified separately and then the ensemble COTS-memory is not separable from test point of view.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

The knowledge acquired in analysing the COTS documentation will allow efficient description of these tests and interpreting their results.

The categories of tests of interest that are implemented are in general:

- Functional testing;
- Endurance testing;
- Detection and Mitigation mechanism testing.

9.2.3.1 Functional testing

Some engineering functional test may be realised in order to guarantee that the characteristics of the COTS requested in the specifications have been achieved ([4] §6.2.1 Objective 1). The COTS is given input data, which adequately characterises the normally expected operation. The outputs are observed and their responses are compared with that given by the specification with respect to:

- Their values,
- The execution time.

Non-compliance with the specification and indications of an incomplete specification are documented.

Some functional tests associated to the behaviour of standardized IP can be delegated to COTS manufacturer. Standardized interfaces like ARINC 429, MIL-STD-1553, PCI or PCIe are particularly adapted to this process because compliance of the interface to the standard can be unambiguously proven by successful realization of a set of test. It can be noted that, for instance, PCI-SIG provides a set of compliance tests [68] in order to ensure compliance of an IP PCIe to the standard. This delegation required minimum confidence gained for instance through assessment of [5] chapter 9 activities [3], [6] and [9].

Even in this case, functional testing may be performed by the COTS customer as they provide elements for good integration of the IP within the COTS and the reference for more elaborated tests with more PoC and more PoO.

Even if PoC and PoO are necessary in interface with the COTS I/O, buried block internal interfaces are addressed through these tests. In this case however the test coverage can be partial because it remains always internal degrees of freedom that can neither be constrained nor controllable.

Consider for instance, a test on a MCU for which some data are prepared in a DDR memory (see Figure 96) through a PCIe port. Consider then that the test protocol consists in testing DMA transfer from the DDR to another PCIe. Results are captured on an external device through PCIe output.

This test does not activate only PCIe and DMA but also the switch matrix, Interconnect, DDR Controller (plus L3 cache on a Freescale Multicore), IOMMU (not represented on the diagram), DDRx itself.

It is not fully clear if all the modes of buried blocks are covered in this test. No PoC can be implemented here so it remains always more degree of freedom on a block like interconnect than constraints to resolve them. It is also the case for DDR controller because of the non-observability of the bus between DDR controller and the DDR itself (see section 5.3.3 for discussion).

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

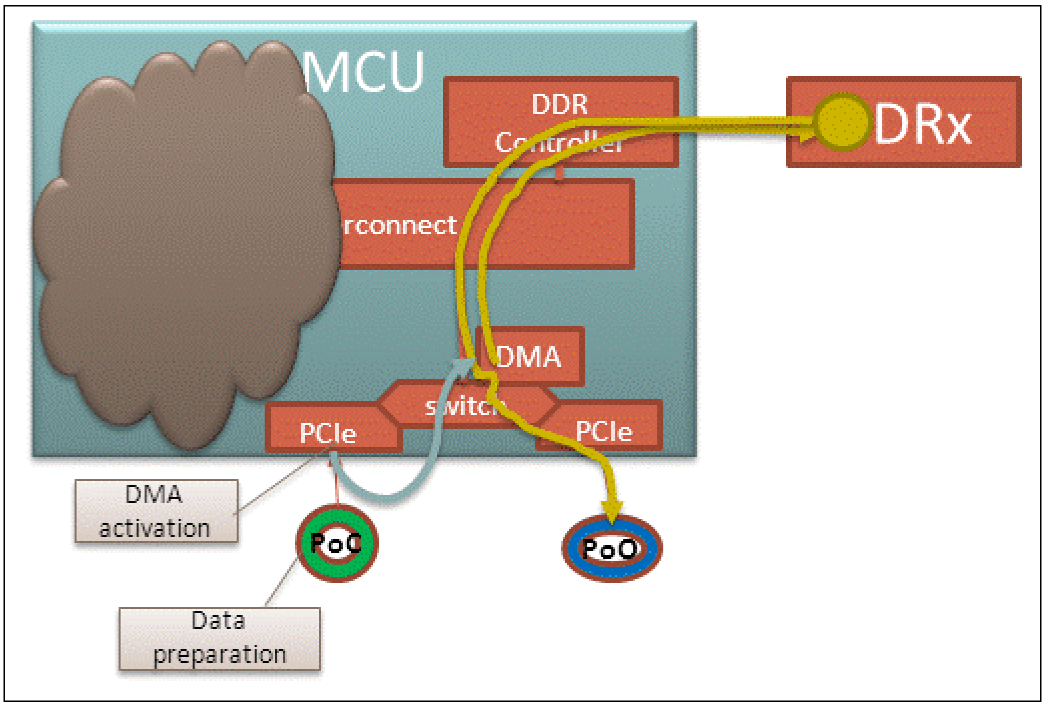


Figure 96: DMA transfer between DDR to PCIe (the DMA->DDR arrow symbolise the read request)

Functional tests are usually extended toward worst case tests for functional conditions. In these tests combination of worst input conditions are generated on IP that can initiate transactions (via PoC) so that the internal IPs are stressed. Such tests aim at finding contentions that could entail the performances of the COTS or its determinism. For instance in the test described here before (Figure 96), increasing volume of data are transferred, with other resources of the COTS soliciting the common internal blocks such as interconnect. In this condition the PoO checks if data are not slowed-down, lost or corrupted.

9.2.3.2 Endurance testing

Endurance tests aim at testing the hardware element with input data selected in accordance with as many dynamical configuration as possible. We mean here by dynamical configuration scenarios of interaction of the COTS with its environment.

It appears that the randomization of test cases and scenario (See Dervin 2012: [69]) added with a systematic definition of test classes with multiple PoC and PoO that this class of tests can detect COTS design errors. In such a test different scenario (corresponding in general to the cyclic permutation of memory cells (similar to a 15-puzzle) are chained in a random way. The histories generated by accumulation of elementary scenarios are close to operational scenarios. Complex scenarios reported in errata sheet are in general reproduced during this test.

Note that extension of functional tests, as expressed on the previous example around Figure 96, lead to the endurance test of internal interfaces.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9.2.3.3 Tracking of failures during functional and endurance tests

Failure may occur during test realisation, in particular during endurance tests that are specified with objective to reveal design failures.

Some "in-test" monitoring has to be specified and realized in order to check the non-occurrence of these failure modes:

- "loss of data", "untimely or forbidden transition of information" and "impossible transition of information" can be detected through comparison of data received with data expected;
- "untimely transfer of message" and "abnormal sequence of messages" can be detected by numbering the transactions;

It is in general difficult to perform these checks on data exchanged for all the data. During a test campaign a compromise has to be reached between the lengths of histories tested (operational efficiency of the test) and the granularity of check performed on data exchanged (relevancy of the tests).

These different topics are summarized in (COTS-AEH_Suggestion_2.):

COTS-AEH_Suggestion_2.	<u>Verification of information transmitted during tests</u> During test sequences, data transmitted through the COTS should be monitored with respect to the possible failure mode identified.

Two other particular types of data should in addition be monitored in order to track design errors:

(1) Verification that embedded error detection and mitigation mechanisms do not mask any design error.

During test, monitoring of embedded mechanisms allows verifying that they do not hide any malfunction. For instance,

- Retry mechanism of PCIe could hide a loss of transaction by physical layers. Monitoring this mechanism allows checking this loss.
- An ECC on a memory could correct errors in read or write transactions. Monitoring the status of ECC allows checking this untimely or forbidden transition.

When this status monitoring is difficult to perform, temptation could be great of deactivating the corresponding mechanism during test so that the result of the test could be negative in case of error. Such a solution should be advanced with caution. Indeed, deactivating during test, a mechanism that will be in place in operation, changes the studied device under test and weakens the conclusions of the test.

On (highly) complex COTS monitoring of embedded mechanisms is in general possible through access to status registers either directly or through PIC entries.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

This important point is formalized in (COTS-AEH_Suggestion_3.);

COTS-AEH_Suggestion_3.	Verification of detection / mitigation mechanisms status during tests During test sequences, COTS internal detection/Mitigation Mechanisms that are embedded in COTS or in COTS interfaces should be monitored and their status should be reported

(2) Detection of unexpected activation of deactivated features can be done by monitoring of their deactivation registers and / or monitoring of their outputs.

During, both functional and endurance testings, features that have been deactivated by configuration could unexpectedly reactivate.
This could be probably due to a bit flip in configuration registers but maybe also due to a degraded functioning caused by an influence of another block at physical level even if the configuration bit is still at deactivated value (Figure 97).

In order to cover these two causes, it is important to monitor the deactivated features by two means:

- The configuration - this can be done at the end of each test bench;
- The activation of inhibited features – this can be done by output monitoring.

COTS-AEH_Suggestion_4.	Status verification of inhibited functions during tests During test sequences, both configuration status and outputs of COTS inhibited features should be monitored.

If during testing, no unexpected activation was noted without bit flip in register, then it appears that register monitoring is an admissible detection mechanism in order to prevent unexpected activation of deactivated features. This mechanism will be described in section 9.3.4.2.

In order to be able to compare the different test results the following suggestion is proposed:

COTS-AEH_Suggestion_5.	Configuration management of COTS under test When test are performed on several instances of the same COTS: (a) their configuration should be identical; (b) This configuration should be managed in configuration; (c) An impact analysis should be performed in case of modification of configuration during project time (after exploitable tests begun).

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

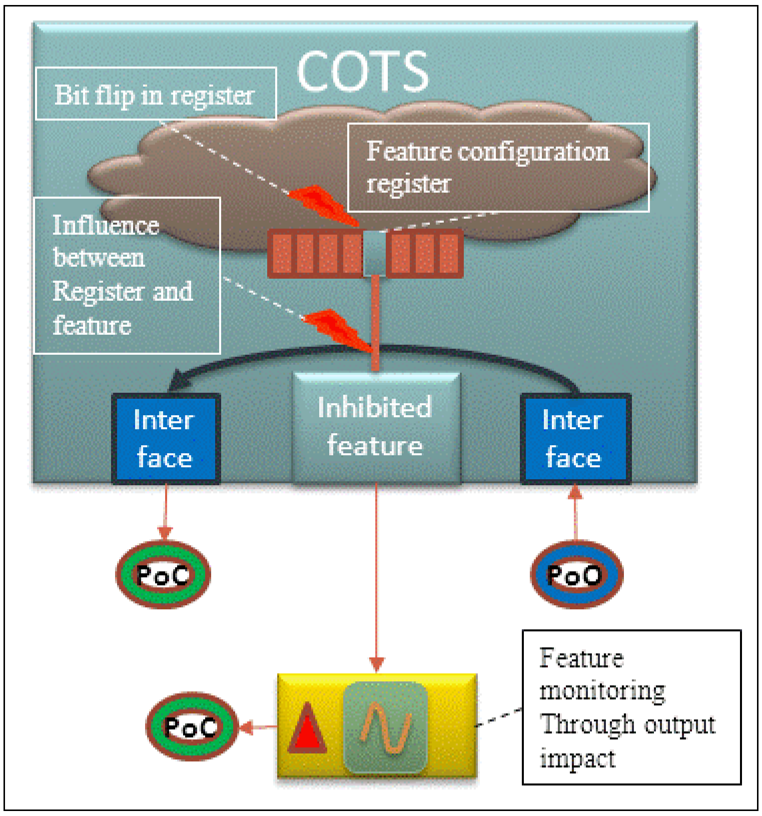

COTS
Bit flip in register
Feature configuration register
Influence between Register and feature
Inter face
Inhibited feature
Inter face
PoC
PoO
PoC
Feature monitoring Through output impact

Figure 97: Unexpected activation of a deactivated feature.

Due to the volume of data exchanged during tests (that are in general automatized) the compromise between the number of tests performed and the data collected during tests is also applicable (see beginning of this subsection). Full monitoring of configuration registers and of output characteristics cannot be applied directly and two possibilities are offered for detection of possible “loss of data”, “untimely” and “impossible transition of information”:

- Compute a CRC on data in addition to the cyclic permutation operation;
- Compare only a sample of data to the before and after test.

9.2.3.4 Test of detection and mitigation mechanisms

In subchapter 9.3 various mechanisms are described and classified. The correct functioning of these mechanisms has to be assessed by test.

COTS-AEH_Suggestion_6.	Verification of Detection and Mitigation Mechanisms During integration tests at various levels, detection / mitigation mechanisms should be tested in particular through fault injection tests.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

The test procedure dedicated to this kind of test is fault injection. Fault injection aims at verifying that detection and mitigation mechanisms trigger correctly when a fault in injected in the device. They can be assimilated to robustness tests of [4] applied to detection / mitigation mechanisms. Five different types of tests based on fault injection tests can be identified [70]:

- Basic fault injection;
- Out of backup state when fault disappear;
- Non triggering under environmental or operational[56] stress;
- Fault injection under environmental or operational stress;
- Out of backup state when fault disappear under environmental or operational stress.

Basic Fault injection:

- Initialisation: the device is configured so that the mechanism can be activated;
- Triggering event: erroneous input or internal fault;
  Such an error simulated in the data processed by the mechanisms is not always easy to generate. COTS integrated detection/mitigation mechanisms are of two types.
  - First ones correspond to interface related mechanisms (e.g. PCIe ECRC, LCRC, Retry, MIL-STD-1553 Parity, etc. based on the interface standard and its implementation);
  - Second ones correspond to particular protections of buried blocks (Parity and ECC on various caches in microcontrollers).

  The first category of mechanisms can be tested through fault injection along the interface. For instance, it is quite easy with a generator of PCIe frame to generate an erroneous frame along the PCIe Link.
  The second ones integrate in general a check mechanism. It is in particular the case of ECC and parity mechanisms of Freescale microcontrollers that embed a mechanism simulating failures in caches in order to verify the correct functioning of ECC or parity both from configuration and implementation viewpoints. Such a mechanism does not exist for all COTS;
- Observable state: the mechanism should detect the failure and let the device falling in backup mode.

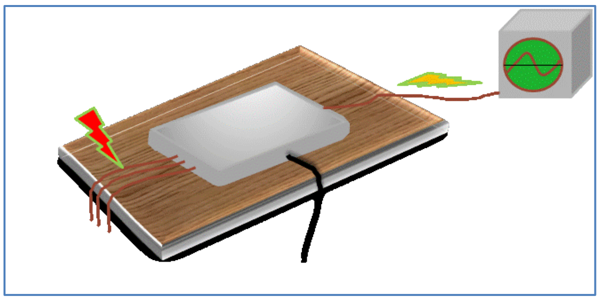


Figure 98: Basic fault injection : when input stimulus go from OK (green) to NOK (red), the output should transit from Ok (green) to backup mode (orange)

[56] Here, environmental will be extended to contextual situation. First order limit conditions for the studied mechanisms are more the task ensures by other part of the COTS than physical environmental conditions albeit EM field should be also considered.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Out of backup state when fault disappear:

- Initialisation: the device in a backup mode as the mechanism triggered;
- Triggering event: Stop of the fault injection;
- Observable state: the mechanism should let the device go out the backup state.

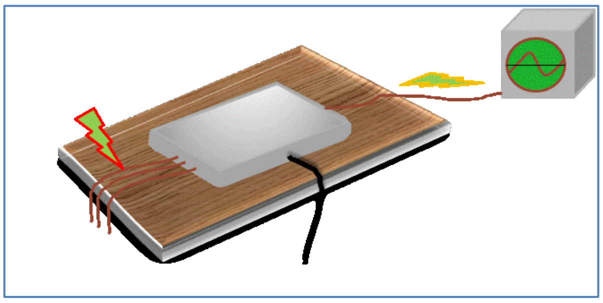


Figure 99: Out of backup state: when input stimulus goes from NOK to Ok, the output should transit from backup to Ok

Non-triggering under environmental or operational stress:

- Initialisation: the device is configured as in the basic fault injection but it operates under stress. For instance:
  - Under Electro-Magnetic field;
  - Under important workload;
- Triggering event: non faulty input and no internal fault;
- Observable state: the mechanism should let the device in normal mode. In the case of stress due to abnormal workload, this is obtained by monitoring internal mitigation mechanisms during functional and endurance test.

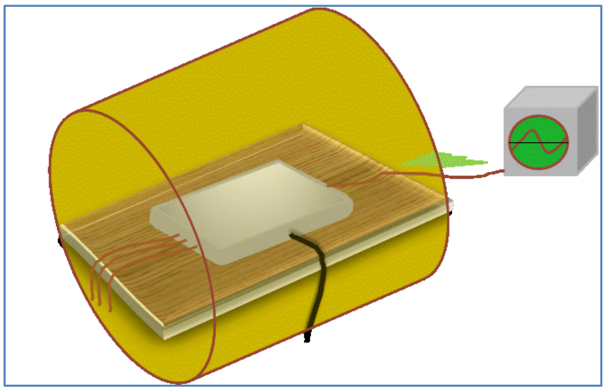


Figure 100: Functional test under stress: the output signal should remain Ok and not transit to backup state.

Fault injection under environmental or operational stress:

- Initialisation: the device is configured as in the basic fault injection but it operates under stress. For instance
  - Under Electro-Magnetic field;
  - Under important workload;
- Triggering event: erroneous input or internal fault;
- Observable state: the mechanism should detect the failure and let the device falling in backup mode. Due to environmental and operational stress the mechanism could react lately or not react at all.

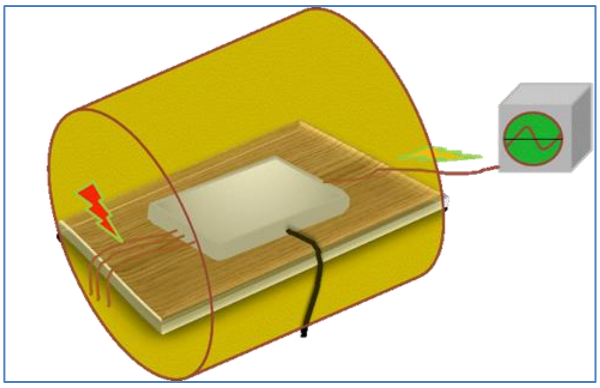


Figure 101: Fault injection under stress: when input stimulus goes from OK to NOK, the output should transit from Ok to backup mode

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Out of backup state when fault disappear under environmental or operational stress:

- Initialisation: the device is configured as in a backup mode and operates under stress. For instance:
  - Under Electro-Magnetic field;
  - Under important workload;
- Triggering event: Stop of the fault injection;
- Observable state: the mechanism should let the device go out the backup state. Due to environmental or operational stress, the mechanism could let the device in the backup mode.

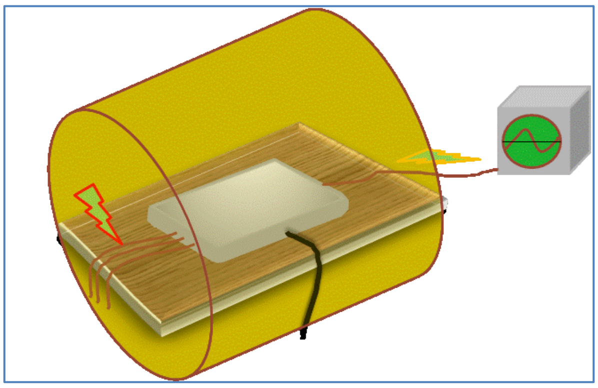


Figure 102: Out of backup state under stress: when input stimulus goes from NOk to Ok, the output should transit from backup to Ok

9.2.4 Test results and constraints on COTS specification

The tests exploitation strategy can be expressed following the process of Figure 103 (Process for detection and mitigation definition) that details the overall process from Figure 94 (General mitigation process.).

This process is organized around COTS specification, test specification and test result exploitation. On the basis of test results the following cases are imaginable:

- **An error rises during test:**
  Rising of an Error during test triggers an analysis that determines its possible causes in COTS design or possibly in test specification.
  In case of confirmed COTS error, the COTS specification is modified in order to take into account workaround solutions against design errors discovered during tests.

  Workaround can be either local (at COTS level) or at system[57] level and realized through:
  - Feature complete deactivation (local),
    - Case of a feature not mandatory to user needs;
  - Feature particular configuration (local),
  - Limitation of COTS features through Operating System (system),
  - Limitation of COTS features through hardware (system).

  This workaround definition impacts system design and COTS specification(s).

  If no admissible workaround can be found, COTS use should be reassessed and a backup solution should be researched.

[57] System has to be taken here, in a non-contextual significance: a reunion of some hardware and software.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

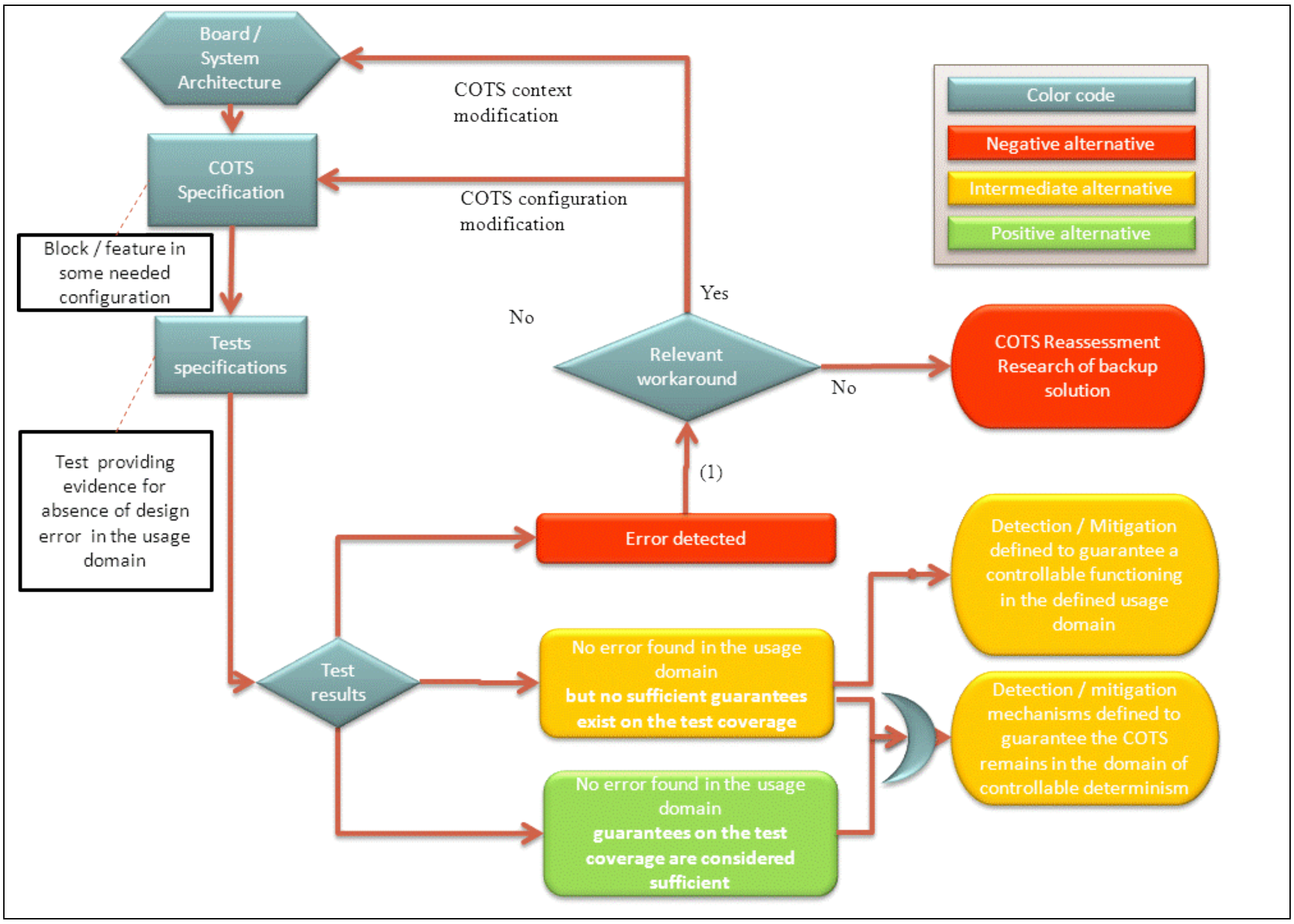

Board / System Architecture
COTS context modification
COTS Specification
COTS configuration modification
Block / feature in some needed configuration
Tests specifications
Test providing evidence for absence of design error in the usage domain
Color code
Negative alternative
Intermediate alternative
Positive alternative
Yes
No
Relevant workaround
No
COTS Reassessment Research of backup solution
(1)
Error detected
Test results
No error found in the usage domain but no sufficient guarantees exist on the test coverage
No error found in the usage domain guarantees on the test coverage are considered sufficient
Detection / Mitigation defined to guarantee a controllable functioning in the defined usage domain
Detection / mitigation mechanisms defined to guarantee the COTS remains in the domain of controllable determinism

Figure 103: Process for detection and mitigation definition

- **No error found in the usage domain.**
  In this case the question of coverage by test of COTS possible usages and behaviours should be asked.
  - If this coverage is not considered as sufficient then two types of error detection and mitigation should be defined.
    - Detection / Mitigation defined to guarantee a controllable functioning in the defined usage domain;
    - Detection / mitigation mechanisms defined to guarantee the COTS remains in the domain of controllable determinism.
  - If this coverage is considered as sufficient (for instance for a COTS with simple peripheral linked by a simple and well known bus) then the detection and mitigation mechanisms to be designed will be limited to those that guarantee the COTS remains in its defined usage domain

At this stage coverage cannot be proven mathematically. Engineering judgment will be used to establish the separation between cases for which the test qualitative coverage is considered sufficient, from those for which it is not.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE TECHNICAL LIBRARY ABBOTTAEROSPACE.COM

The complete mitigation set consists in

- Guaranty of a safe and deterministic perimeter,
- Controllability of failures from zones none fully testable.

It is organize in two main set of methods

- General methods that principle is in great part independent from the COTS studied and can even cover several COTS
- Particular methods applicable to one COTS and that can rely on internal diagnostics of this COTS.

9.3 MECHANISMS FOR DETECTION AND MITIGATION OF DESIGN ERRORS

9.3.1 Introduction

COTS design errors mitigation can be understand at several levels of breakdown described hereafter from the deeper to the wider (see Figure 104):

- <u>COTS level</u>: some mitigation means are embedded within the COTS. They have been described in the previous chapters and exploited in a forthcoming section;

- <u>Board level</u>: it is the principal level on which the architectural mitigation means takes place;

- <u>LRU (Line Replaceable Unit) level</u>: by extension of the board it can contain some mitigation means even if in general by design it should be possible to favour the board level for locality and readability reasons;

- <u>Avionic suite and aircraft level</u>: Even if as a last resort, functional safety mechanisms at aircraft level could detect and mitigate a COTS design error that could have leak from LRU level, it should be ensured that the mitigation should take place at the next higher level of COTS component in the equipment/LRU breakdown or at least at the closest higher level as possible. Indeed, the objective is to ensure that the Development Assurance Level assigned to an equipment/LRU is achieved with the expected rigor.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

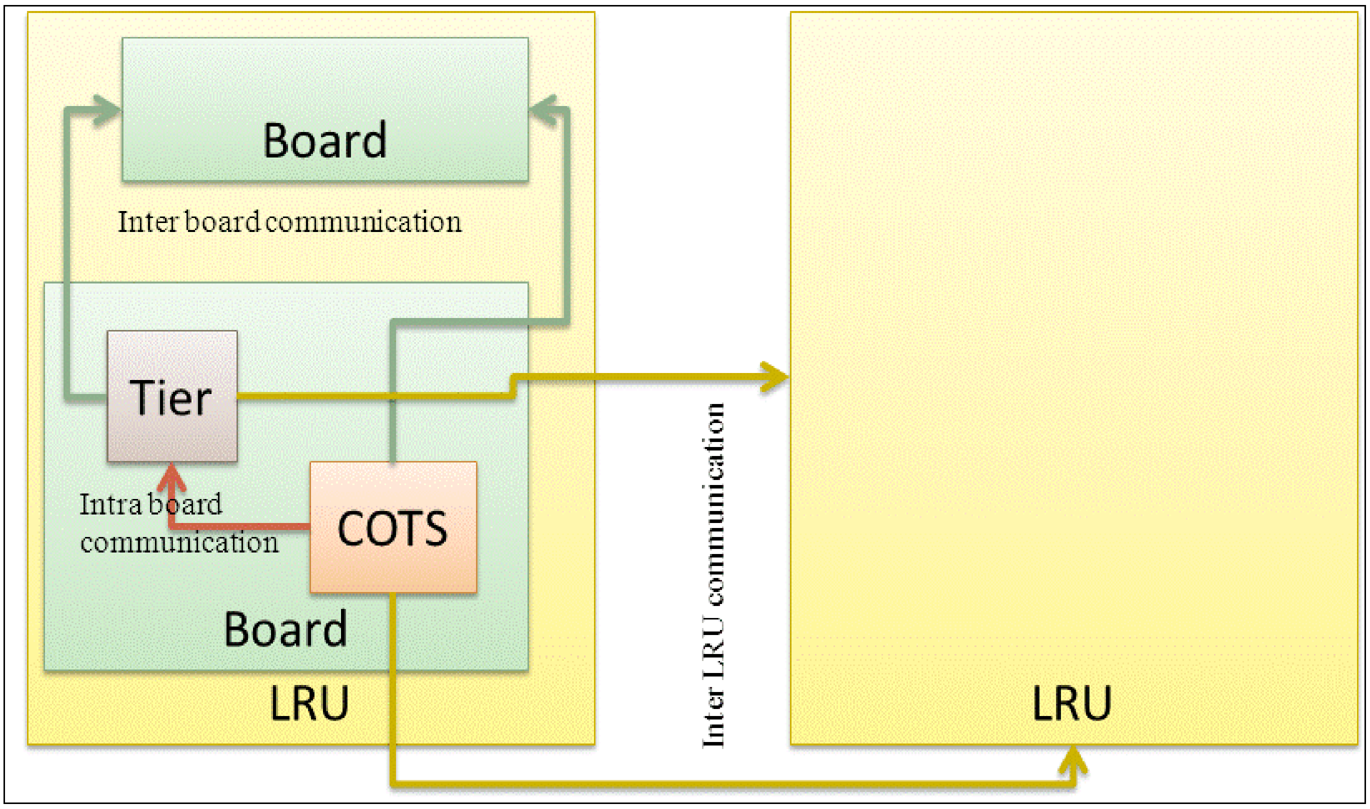


Figure 104: different levels of locality in comunications

This could be summarised in the following Suggestion (COTS-AEH_Suggestion_7.)

COTS-AEH_Suggestion_7.	<u>Integration level for definition of detection / mitigation mechanisms</u> During definition of detection / mitigation strategy, COTS design error detection / mitigation mechanisms should be defined as closer as possible from the COTS. System and aircraft levels may be considered only when no detection / mitigation could be implemented locally.

This suggestion led us to concentrate more on local mechanisms than on aircraft architectural mechanisms. The different mitigation means relevant to our goals are described hereafter. For complete reviews, please refer to IEC-61508 [26] part 7 or IS0-26262 [9] part 5 Annex D.

Study of the available mechanisms split them in three categories:

- Mechanisms relying on internal mitigation means of the COTS; These mechanisms have been described in the corresponding sections of chapters 7 and 8. These mechanisms are grouped in a dedicated section hereafter (9.3.2);
- Mechanisms relying on internal COTS detection means and on external mitigation means (Mixed mechanisms covered in section 9.3.3). These mechanisms are shared among internal COTS resources and architectural means. This mechanisms use in general PIC and debug interfaces;
- Mechanisms fully relying on architecture means for detection and mitigation are covered in sections 9.3.4.1 to 9.3.4.5.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9.3.2 Internal failure detection / mitigation mechanisms

Following table summarizes the detection / mitigation mechanisms described in chapter 8.

Carrier block	Mechanism	Failure covered	Blocks covered by mechanisms	Applicable to COTS type	Reference Section for mechanism
NAND Flash card ECC block	ECC	○ Untimely or forbidden transition of data ○ Impossible transition of data	NAND Flash memory array	○ NAND Flash	8.6.5
Cores	Parity and ECC	○ Untimely or forbidden transition of information ○ Impossible transition of information	○ Core (caches) ○ Various other memories internal to microcontrollers	○ Microcontrollers ○ Multicores	8.7.3.1.3 8.8.3.1.3.1 8.8.3.1.3.2 8.8.3.3.3 8.8.3.4.3
Cores	MMU	○ forbidden transition of address	Cores	○ Microcontrollers ○ Multicores	8.7.3.1.3 8.8.3.1.3.3
Cores	Embedded hypervisor	○ Untimely or forbidden transition of address ○ Impossible transition of address	Cores	○ Multicores	8.8.3.1.3.4
MPX bus	Detection of data and address tenure termination and Retry request	○ Loss of messages	Cores All peripheral blocks connected	○ Microcontroller	8.7.3.2.3
MPX bus	Address and data parity	○ Untimely or forbidden transition of information ○ Impossible transition of	Cores All peripheral blocks connected	○ Microcontroller	8.7.3.2.3

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Carrier block	Mechanism	Failure covered	Blocks covered by mechanisms	Applicable to COTS type	Reference Section for mechanism
		information			
OCeaN™	Packet transfer termination detection (potential)	○ Loss of messages	○ Connected Peripherals ○ DMA	○ Microcontrollers ○ Multicores	8.7.3.4.3
CoreNet ™	coherency violation and local access error detection	○ Untimely or forbidden transition of address ○ Impossible transition of address	○ Initiators connected to CoreNet ™ ○ CoreNet ™ (partially)	○ Multicores	8.8.3.2.3
CoreNet ™	Retry Mechanism	○ Loss of message	○ Initiators connected to CoreNet ™ ○ CoreNet ™ (partially)	○ Multicores	8.8.3.2.3
CoreNet ™	Transaction ordering mechanism	○ Abnormal sequence of message	○ CoreNet ™	○ Multicores	8.8.3.2.3
CoreNet ™	ECC on buffers	○ Untimely or forbidden transition of information ○ Impossible transition of information	○ CoreNet ™	○ Multicores	8.8.3.2.3
PAMU	Detection and avoidance of forbidden access of peripheral to CoreNet ™	○ Untimely or forbidden transition of address ○ Impossible transition of address	All Peripherals connected to a PAMU: ○ DMA, ○ OCeaN™, ○ PCIe, etc.	○ Multicores	8.8.3.3.3

Table 19: summary of COTS internal detection/mitigation mechanisms;

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Note: the ECC and parity on DDR memories cannot be considered strictly as internal failure detection and mitigation means as DDR controller detect and correct errors from DDR DRAM chip. It is thus an architectural mechanism. However, as outlined in section 5.3.3.1, it is not possible to define observation point between DDR controller and DDR DRAM chip, the system DDR controller and DDR DRAM chip is strongly coupled and can quasi be considered as a single system.

COTS internal detection / mitigation mechanisms can be used in the global strategy defined to cover COTS design errors. In order to rely on these mechanisms, few conditions are identified in the following suggestion, based on, sections 9.2.3 and 9.3.5:

COTS-AEH_Suggestion_8.	<u>Usage of COTS internal detection/mitigation mechanisms</u> During COTS design error mitigation strategy elaboration, a detection / mitigation mechanism implemented within the COTS can be selected if ○ Its triggering can be monitored during COTS functional and endurance tests; ○ It can be tested by fault injections tests on COTS; ○ A mechanism can be implemented in operation in order to cover its latent failures.

9.3.3 Mixed mechanisms

Mixed mechanisms are detection / mitigation mechanisms for which failure detection is performed within the COTS and mitigation is performed by architectural means.

Complex COTS embed different hardware accelerators that detect or concentrates failures, like PIC or JTAG (or NEXUS) modules.

PIC (or MPIC) manages failures generated internally or transmitted via the inputs (PCIe) and controls also the interruption critical or not on requests. In the case of a MCU, a solution is that the MCU delegates to PIC some reset of the COTS. PIC and machine interrupts offer in general powerful possibilities for critical interruptions: reset of one core by the MPIC, reset of one core by the other, reset of one core by itself…

Difficulties rise to use these advance features:

- The complex configuration of these blocks (see subsection 8.8.3.5)
- Asynchronism:
    - Non Maskable critical IT rely in large part on external events so that internal COTS functioning should be disturbed by asynchronous external disturbance,
    - Synchronism has to be ensured at computing platform level. This could not be longer ensured in case of local reset by a MPIC or by a Core machine check.

Due to these reasons the proposed solution consists in the capture of PIC status by an external device (e.g. PLD) that takes decision about the action on the COTS in coherency with the state of the computing platform.

It result from this argument the following type of detection mitigation:

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

Principle
An Interruption Request (IRQ) is received by the MPIC (see for instance Figure 93) and is captured by an external device that takes a sanction in order to mitigate the detected failure.

Failure covered
The internal failures covered depend upon the IRQ raised.

A variant of this mechanism can be operated by monitoring the Debug block. This block offers JTAG or Nexus diagnostic capabilities during development, fabrication or maintenance tests. During operation, it scans the COTS blocks non-intrusively. Its output could be captured by an external device in order to perform fine diagnostics of the COTS and take appropriate sanction.

COTS-AEH_Suggestion_9.	<u>Usage of COTS internal detection mechanisms</u> During COTS design error mitigation strategy elaboration, a detection / mechanism implemented within the COTS (such as PIC or JTAG blocks) can be selected if : o Its triggering can be monitored during COTS functional and endurance tests; o It can be tested by fault injections tests on COTS; o A mechanism can be implemented in operation in order to cover its latent failures. In this case, mitigations should be applied by an external device.

PIC and Debug exploitation are interesting means to manage errors generated internally to the complex COTS, but do not allow full coverage of the internal errors generated. The possibility of common points between the initial error and the detection mean may be also keep in mind.

9.3.4 Architectural mechanisms

9.3.4.1 Monitoring of outputs

Outputs described in Chapter 7 offers possibilities of local monitoring that have to be considered. It is necessary to remain aware that this monitoring cover in general only the lower layers of the interfaces and that they do not give accesses to the buried blocks. This general remark has sometime to be modulated as it is discussed in following subsections. The general model can be depicted as followed (Figure 105). It leads to the following suggestion:

COTS-AEH_Suggestion_10.	<u>Usage of COTS output monitoring</u> During COTS design error mitigation strategy elaboration, if a detection mechanism based on COTS outputs monitoring is used o The detection principle should be specified to the message receiver; o The message receiver monitoring implementation should be tested in integration tests (*); o A mechanism can be implemented in operation in order to cover detection mechanism latent failures. (*) the test can be done at board, LRU or system level, depending on the receiver.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

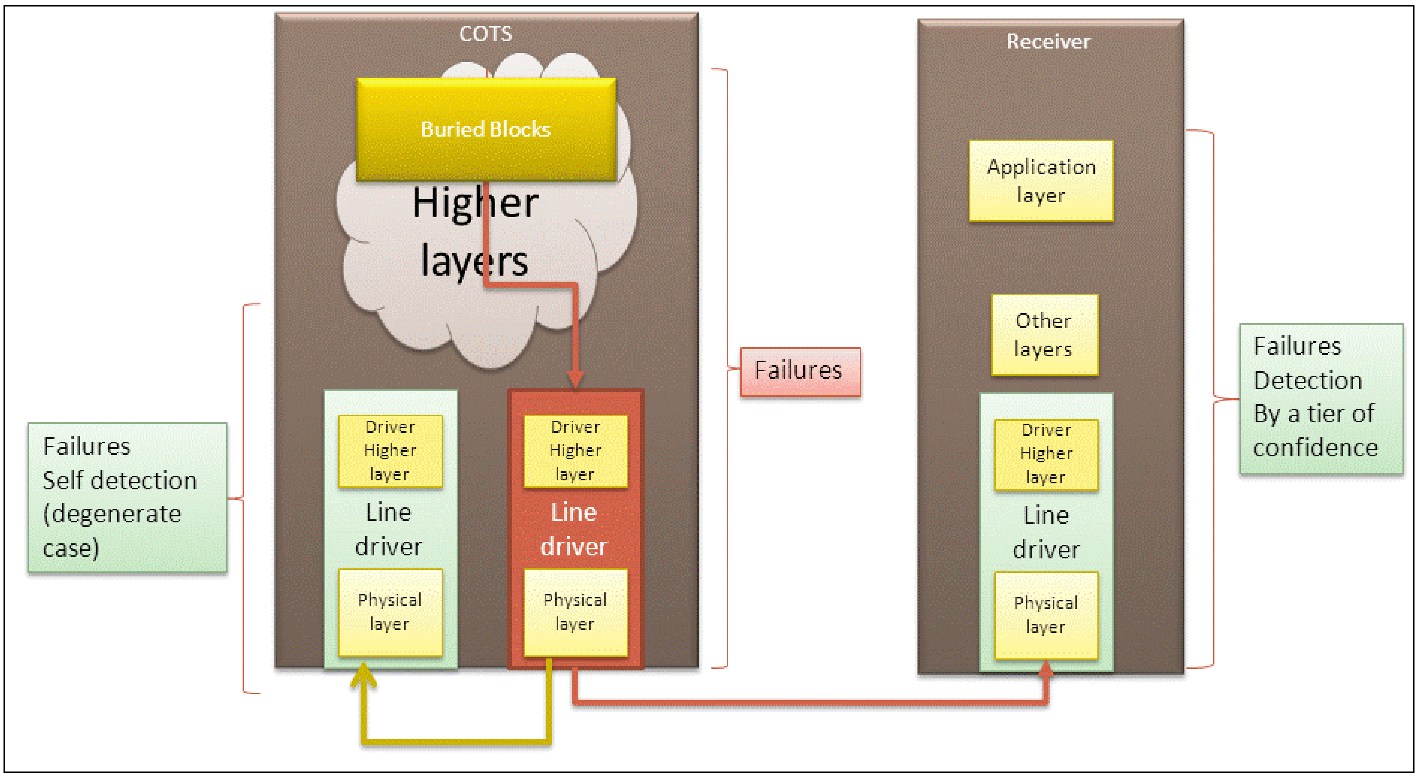


Figure 105: COTS Output monitoring

9.3.4.1.1 Discrete I/O

As already described, a discrete I/O does not embed intrinsically failure mitigation mechanisms.

Some COTS design can implement read back monitoring. In this case the sent value is monitored and stored in a status register (Figure 106). This status register has to be compared to the command register by an internal arbiter (can be the PIC, a core or a Debug module). We are in this case in the “degenerate case” of Figure 105.

Such a mechanism can is not systematically implemented by COTS. When it is so it should be exploited in order to cover:

- Untimely or forbidden transitions,
- Impossible transitions.

Note: As explained in subchapter 7.2 these two modes are the only one admissible for discrete I/O.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

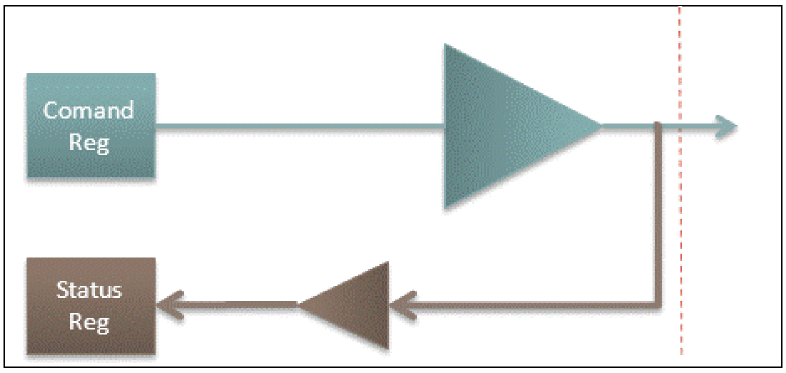


Figure 106: Discrete I/O read back monitoring internal to COTS

When it is not implemented internally it can be created between an output and an input of the COTS, Figure 107 (a), or some redundancy between two I/O, Figure 107 (b).

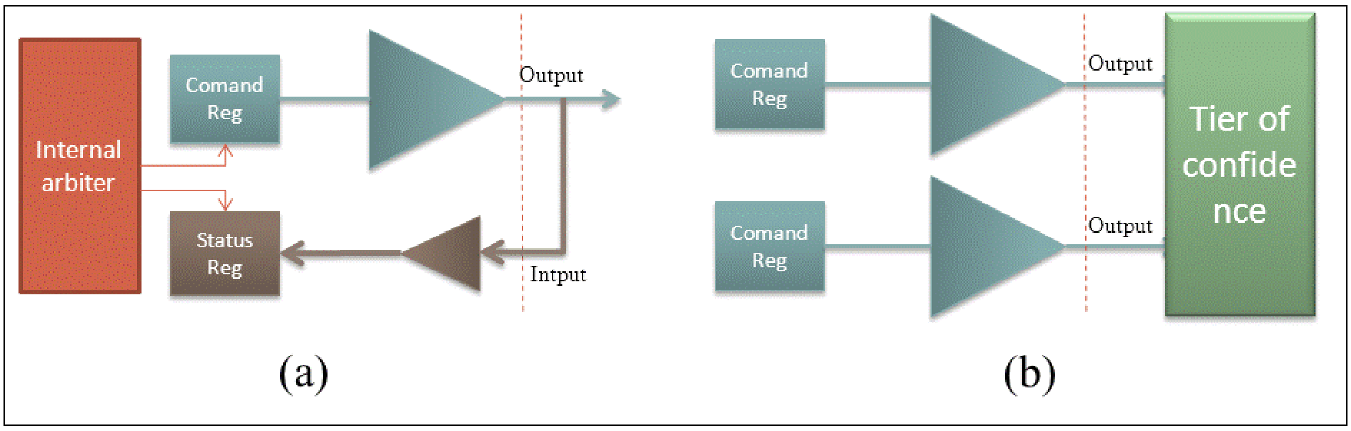


Figure 107: External read back monitoring (a) and redundancy (b) for Discrete I/O

In the redundancy case it is important that the two command registers are filled as independently as possible in order to avoid common point. If it is realized the coverage claimed can largely overtake the Discrete I/O interface block.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

COTS Output	Mechanism	Failure covered	Blocks covered	Section
Discrete	Read back monitoring	untimely transfer of messages Loss of message Untimely transfer of message Untimely or forbidden transition of information Impossible transition of information	Discrete interface block Buried blocks (less buried than the detection block)	7.2
Discrete	Signal redundancy	Loss of message Untimely transfer of information Untimely or forbidden transition of information Impossible transition of information	Discrete interface block Buried blocks (less buried than the last common block)	7.2

Table 20: summary of detection/mitigation mechanisms on discrete output

9.3.4.1.2 Serial Peripheral Interface

Two mechanisms are described in section 7.3.

- Detection of overclock pulses by slaves,
- Parity bit encoding by master.

COTS Output	Mechanism	Failure covered	Blocks covered	Section
SPI	detection of extra CLK pulses (optional)	untimely transfer of messages (partly)	SPI block CLK	7.3
SPI	Parity encoding by master	Untimely or forbidden transition of information Impossible transition of information	SPI block (partially)	7.3

Table 21: summary of detection mechanisms on SPI

Complementary electrical detection mechanisms can be added at physical level. At higher levels end-to end mechanisms (see subsection 9.3.4.3) can be implemented on SPI.

9.3.4.1.3 ARINC 429

ARINC 429 implements physical and data link layers mechanisms that can be exploited in order to monitor the signal generated by a COTS in the framework of output monitoring depicted on Figure 105.
The detection of deviation from characteristics described in section 8.3.4 can be performed:

- At physical layer: electrical, signal shape, signal timing… This monitoring can prevent against some drift that could at the end lead to "loss of information".
- At data link layer: Parity bit;
- At higher layer: some special error detection can be implemented by Operating System or even at applicative layer. In particular a coherency check can be performed between:

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- ○ Sign/Status Matrix (SSM) bit field;
- ○ Source/Destination Identifier bit field;
- ○ Interface Control Document (ICD) predefined label and data format for different emitter types (industry standard).

The following table summarises this analysis:

COTS Output	Mechanism	Failure covered	Blocks covered (of emitter)	Section
ARINC 429	Rejection of physically non conform message	Untimely or forbidden transition of information Impossible transition of information	ARINC 429 block	7.4
ARINC 429	Configurable Parity Bit encoding and check	Untimely or forbidden transition of information Impossible transition of information	ARINC 429 block	7.4
ARINC 429	Detection by high level layers of receiver of incoherency between - Sign / Status Matrix (SSM) bit field - Source/Destination Identifier bit field - ICD predefined label and data format for different emitter types (industry standard).	Untimely or forbidden transition of address Impossible transition of address	ARINC 429 block Buried blocks	7.4

Table 22: summary of detection mechanisms on ARINC 429

9.3.4.1.4 MIL-STD-1553

Embedded mechanisms of MIL-STD-1553 are exploitable in different layers, in the framework of “output monitoring” depicted on Figure 105.

- At physical layer: Detection by the receiver of signal characteristic deviations (NRZ, Manchester structure and characteristic timings), allows detecting word loss situations and untimely, forbidden or impossible transition of information due to physical layers. Higher layers errors are not covered as they do not manipulate electrical encoded signal but bits.

- At Data Link Layer: Section 7.5.4 lists the protocol elements which deviation should be monitored in order to detest an abnormal behaviour from one of the elements.

The Bus Controller remains the master of the bus so that its diagnostic should only be performed by itself. Some Additional mitigation can get round this problem:

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

- MIL-STD1553 accepts the redundancy of BC if and only if only one speaks at each time on the bus. This do not solve the problem of BC error detection;
- It is conceivable to use a "Bus Monitor"[58] as an independent item that could then monitor the health of the communication on the bus and take a sanction on the BC if necessary.

The following table summarises this analysis:

COTS Output	Mechanism	Failure covered	Blocks covered (of emitter)	Section
MIL-STD-1553	Detection of NRZ violation	Loss of message	MIL-STD-1553 block	7.5
MIL-STD-1553	Detection of Manchester encoding violation	Impossible transition of information	MIL-STD-1553 block	7.5
MIL-STD-1553	Redundancy of MIL-STD-1553 lines	Loss of message Untimely transfer of message Abnormal sequence of message Untimely or forbidden transition of information Impossible transition of information	MIL-STD-1553 block More buried blocks	7.5
MIL-STD-1553	Parity bit encoding	Untimely or forbidden transition of information Impossible transition of information	MIL-STD-1553 block	7.5
MIL-STD-1553	Detection of violation of message type or format	Untimely transfer of message Untimely or forbidden transition of information (partially)	MIL-STD-1553 block	7.5
MIL-STD-1553	Detection of violation response time slice	Untimely transfer of message (including advanced responses and babbling)	MIL-STD-1553 block More buried blocks	7.5

Table 23: summary of detection mechanisms on MIL-STD-1553;

9.3.4.1.5 PCI Bus

Embedded mechanisms of PCI (described in section 7.6.3) can be handled at various levels.
At data link layer in particular,

- the Parity can protect against untimely or forbidden transition and impossible transition. However this parity protect only from these modes when they are generated by lower layers (after the encoding of parity and before its decoding);
- the master abort mechanism protect efficiently against loss of messages due to slave error. The master remains critical.

[58] In principle BM is dedicated to performance monitoring of the bus and not to error detection.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

•

The following table summarises this analysis:

COTS Output	Mechanism	Failure covered	Blocks covered (of emitter)	Section
PCI	Parity	Untimely or forbidden transition of information (partially) Impossible transition of information (partially)	PCI block (partially)	7.6
PCI	Detection of delayed answer and Master abort	Loss of message Untimely transfer of message (delayed)	PCI block More buried blocks	7.6

Table 24: summary of detection mechanisms on PCI;

Additional mechanisms can be defined to protect generation and consumption of PCI messages. These mechanisms take place at higher layers and are tackled in next sections.

9.3.4.1.6 PCIe bus

As described in the PCIe subchapter, PCIe is a very robust network with several embedded mechanisms in particular at Data Link and transaction layer. They protect very efficiently from various modes in the lower layers

a. A Sequence number encoded at physical layer (not described in 7.7.4) protect against untimely transfer of message (babbling) and abnormal sequence of messages at physical layer;
b. ECRC protect against untimely, forbidden and impossible transition from emitter transaction layer to receiver transaction layer;
c. LCRC protect against untimely, forbidden and impossible transition from emitter Data Link layer to receiver Data Link layer;
d. Frame Re-transmission and in particular Acknowledgement mechanism prevent against loss of frame from DLL to DLL and also against untimely transfer of messages (delayed);
e. Adjacent Device's Memory Availability and flow management complementary protect against loss of frame ad DLL.
   Adjacent Device's Memory Availability and flow management by managing the internal buffers of receiver PCIe block primarily prevent the loss of message that could be sent to this block while it is busy. Its classification in the present category of mechanisms should be understood in the following sense: internal detection (detection by the COTS that its buffers are near full) and architectural mitigation (mitigation by the emitter by delaying the sending).

In a microcontroller all of these mechanisms status are accessible through PIC that can have some status of health of PCIe connection.
The following table summarises this analysis:

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

COTS Output	Mechanism	Failure covered	Blocks covered (of emitter)	Section
PCIe	Sequence number decoding	Untimely transfer of message (babbling) Impossible transition of information (partially) Abnormal sequence of message	PCIe blocks (partially)	7.7
PCIe	ECRC decoding	Untimely or forbidden transition of information Impossible transition of information	PCIe blocks (up to TL)	7.7
PCIe	LCRC decoding	Untimely or forbidden transition of information Impossible transition of information	PCIe blocks (up to DLL)	7.7
PCIe	Acknowledgement and retry mechanisms	Loss of message Untimely transfer of message (delayed)	PCIe More buried blocks	7.7
PCIe	Detection of receiver buffer over load (flow management)	Loss of message	PCIe	

Table 25: summary of detection mechanisms on PCIe;

9.3.4.1.7 Periodic frame monitoring

In addition to previous monitoring on each output type, it is usual in the case of buses and network with periodic frame, word, transactions identified (exclude SPI) to scan the arrival of frames. Depending upon the admissible latency, the status of loss of frame can be declared after non detection of 1 to 3 frames.

Principle
The frame reception is confirmed on few frames (typical value 3).
If a frame is not received the receiver operates a transition in a wait state with a temporary backup mode (in general the last valid value received).
If after the confirmation time span no frame is received, the applicative layer is informed and is responsible to define a sanction.

Failure modes covered
This mechanism is mainly dedicated to coverage of *loss of messages* failure mode.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

COTS-AEH_Suggestion_11.	Periodic frame monitoring During COTS design error mitigation strategy elaboration, if a periodic frame monitoring mechanism is used in the framework of COTS output monitoring: ○ The latency induced by the confirmation time should be considered.

9.3.4.2 Configuration monitoring

Has already studied in various sections, many COTS configuration registers are very critical. Tests should show that modification of COTS feature behaviours can only be due to configuration change and not to untimely changes due to physical effects without register corruption (consider section 9.2.3.3).

COTS-AEH_Suggestion_12.	COTS configuration monitoring During operation, any change in COTS critical configuration registers should be detected by a periodical monitoring. This monitoring may be tested in order to avoid latent faults except if the default configuration of unused blocks or features is showed to be innocuous.

A classical way of doing is to store a complemented to 0 value of the useful configuration (mirror configuration) in memory and periodically check that the sum of the configuration and of the mirror configuration give 0.

This can be done by the COTS when it has computation resources (microcontroller) or by an independent item when the COTS has no computation resources (e.g. a bridge). In the first case, if the configuration check is realized by the microcontroller itself, the mechanisms should be complemented by a health monitoring of the microcontroller in order to avoid that an untimely or forbidden change of the MCU configuration stop every capabilities to check the configuration.

If such a complementary mechanism cannot be in place in a fault tolerant time interval delay then the monitoring of configuration by an independent item (see subsection 9.3.4.4) could be preferred.

9.3.4.3 End to End protection

9.3.4.3.1 Principle

End to end protection aims at protecting data exchanged from higher level in the emitter to higher level in the receiver. This encoding can be performed by the Operating System or by the application. It is applicable to the exchange of data computed by a microcontroller and another item that can be either a microcontroller or a PLD.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Consider the communication between to applications as represented on Figure 108.

Depending upon the type of failure to cover, end-to-end protection can encode on each transaction at highest possible level (Figure 108):

- CRC,
- process counter,
- Dating for non-periodic transactions.

9.3.4.3.2 Detailed presentation

Encode/decode in higher layers covers the different internal block that are crossed by the transaction. Moreover, a bridge (for instance) situated between the emitter and the receivers is also covered by the mechanism.

- An information signature (for instance a CRC) guarantees that the information covered are not corrupted when they cross the different blocks.
- The Process counter labelled each transaction (modulo an integer *n*) and thus guarantees that these transactions are present in the right order in the receiver.
- Dating can be added if untimely transfer of message failure mode has not been excluded by other means.
    - For periotic transactions that are waited by the receiver in certain time slice, it is easy to determine if there are too many transactions (babbling) or if the transaction is late (delayed). In this case the [2]dating of transactions is useless;
    - For non-periodic transaction that can be transmitted anytime, the dating allows knowing at which time it has left the top layer of emitter and if it has been delayed. This is of particular importance in the case of bridges crossing that can delay transactions in some buffers.

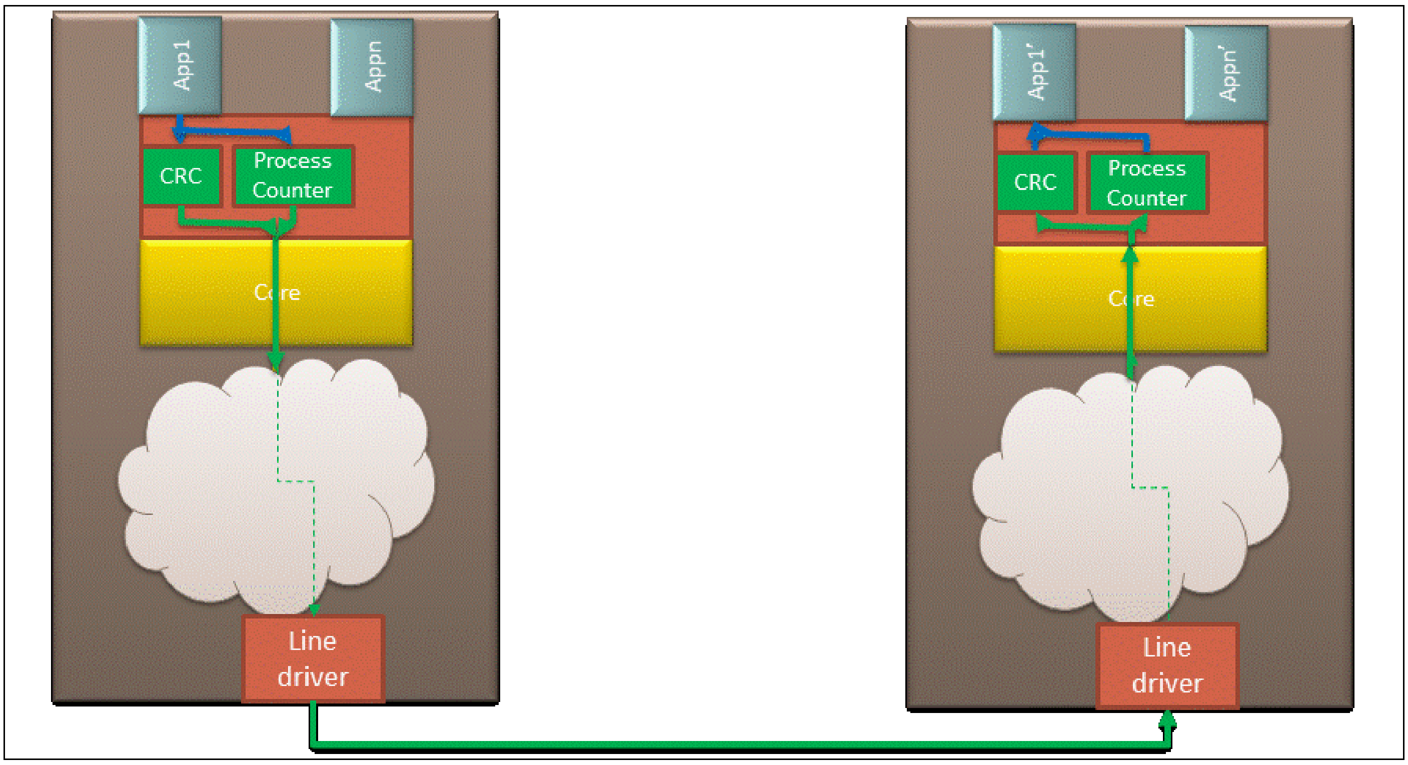


Figure 108: Principle of end-to-end protection (optional dating has not been represented).

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

This mechanism is particularly efficient. In its refined version (CRC, process counter, dating) presented here it deals with difficulties for tuning of dating as its exploitation request synchronization between the emitter and the receiver. This synchronisation is difficult to realize.

It has to be noted that end-to-end protection, despite of its efficiency, is not local to a computing platform and thus does not respect the proposal of COTS-AEH_Suggestion_7. Indeed considering the communications depicted on Figure 104, the COTS is implicated also on inter LRU communication. A full end-to-end encodes protections of the message at operating system level in a part of a message that have to be decoded only by the final receiver. This final receiver is for a majority of messages of interest on another LRU.

Despite of this status, end-to-end protection should be considered in complement to local mechanisms.
There are two way to use weakened local versions of end-to end, both with advantages:

- Local end-to-end: Encode protection at a level that is decoded by a local device. This protection is of the type of LCRC encoded in DLLP of PCIe. The local receiver can be the next device on the line or a more distant device, if a refined encapsulation is defined. For instance, in this case, this mechanism can cross a bridge and cover very efficiently its failures. Nevertheless, this mechanism covers fewer failures than the pure end-to-end.
- Sampled end-to-end: Generate special frames with real data sent periodically to a local device address and encoded from end-to-end. In this case it is very easy to address a particular device that is not necessarily the next one on the line. The coverage of this mechanism is fully satisfactory like pure end-to-end but the protected sample is not 100%.

Note that the three variants (full End-to-end, local end-to-end and sampled end-to-end) are not exclusive and can be implemented simultaneously.

COTS-AEH_Suggestion_13.	<u>End-to-End protection</u> During COTS design error mitigation strategy elaboration, if an end-to-end protection mechanism is defined in order to detect COTS design error, o It should be encoded in COTS higher layers (applicative or higher Operating system layers)

End-to-end protection has been extensively use in CAN communication in automotive world.
When applied at sub-applicative (operating system) on all information of all messages, this mechanism impacts performances. This impact should be determined.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9.3.4.3.3 Failure modes covered

The status of such a mechanism on the basic failure modes use in this report is detailed in Table 26

Failure mode	Sub class of failure mode	Mitigation argument
Loss of message		The process counter detect the loss of m<n consecutive transactions Dating or periodic frame detection allows detecting message loss.
Untimely transfer of message	Babbling Delayed message	By time slice in the case of periodic transactions and by dating for non-periodic transactions. The Process counter can detect babbling caused by lower layers.
Abnormal sequence of message		The process counter detects the abnormal sequence of an arbitrary number of transactions except the cyclic permutation modulo *n*.
Untimely or forbidden transition	Untimely transition of data Untimely transition of address (emitter or receiver) Forbidden transition of address (emitter or receiver)	Covered by the data and address signature
Impossible transition	Impossible transition of data Impossible transition of addresses	Primarily covered by the process counter Covered by the data and address signature

Table 26: coverage of end-to-end protection

9.3.4.4 External independent monitoring on the data path

9.3.4.4.1 Principle

The global principle of monitoring described here is applicable between a MCU or a Multicore and an external item independent from this microcontroller.
The basic principle of this detection/ mitigation mechanism is for the external item to challenge the microcontroller with some question and to compare the microcontroller answer with its own answer. In case of discrepancy between both answers, the external independent item applies a sanction to the microcontroller. In order to be efficient, the monitoring should be implemented on the operational data path.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

9.3.4.4.2 Detailed presentation

In more details, the detection and mitigation mechanism is realised by a monitoring partition (MON) of the MCU. When running,

(1) the MON partition solicited the external independent item that store a test pattern,
(2) the external independent item deliver the data and on its side perform the calculation[59],
(3) MON activate various resources of the COTS (PCIe, DMA, Interconnect, L3, DDR controller, DDR) in order to store in test data in DDR,
(4) MON get the data in DDR and perform the computation,
(5) MON release the result to the external independent item,
(6) the external independent item monitors the arrival of MCU result, compare its own results with MCU result and
(7) the external independent item takes adapted sanction.

In this process, the test pattern is of crucial importance. Its choice can strongly influence the coverage of the global mechanism. It is proposed to implement several tests patterns with different types of instruction to perform and different data type to process.

Several variants of this mechanism may be defined.

- Previous principle can be applied to the monitoring of complex COTS (e.g. bridges) without computation capabilities in different cases
  - The mitigated COTS is on the data path between the microcontroller and the independent item (see the block “Other COTS (e.g. Bridge)” on Figure 109);
  - the monitored device could be inserted on the data path between two independent items or
  - the test patterns could be adapted to some automatic response request (such as posted message in PCIe).
- Application to Multicore processors offer two possibilities:
  - Simple duplication of the error detection / mitigation loop. In this case MON$_n$ running on Core$_n$ realises the previous process independently of any context. Test data are refreshed between MON$_n$ and MON$_{n'}$. Computation resources of the independent item have to be checked, avoiding conflict when two MON request it at the same time.
  - Coherent duplication of the loop. In this case the independent item keeps the same test data in order for each MON partition to use the same data set. The independent item can then compare its result with the n core results and take adapted sanction that could be a reset of the complete platform when all the cores agree against it.
- The independent item can be a COTS so that the mechanisms is implemented between two COTS (A and B) with crossed mechanisms (A solicit B and B solicit A). In this case however the sanctions scheme as to avoid paradox of Byzantine generals and should be study carefully. This can be solved if one of the COTS is simpler that the other one.

[59] It is possible that the test pattern result is already stored within the tier of confidence.

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

COTS-AEH_Suggestion_14.	<u>Monitoring by an external independent item on the data path</u> During COTS design error mitigation strategy elaboration, if a monitoring by an external independent monitoring on the data path is defined in order to detect COTS design error, ○ The data and applied function should be chosen carefully; ○ The periodicity of the monitoring should be considered according to the fault tolerance time interval of the system.

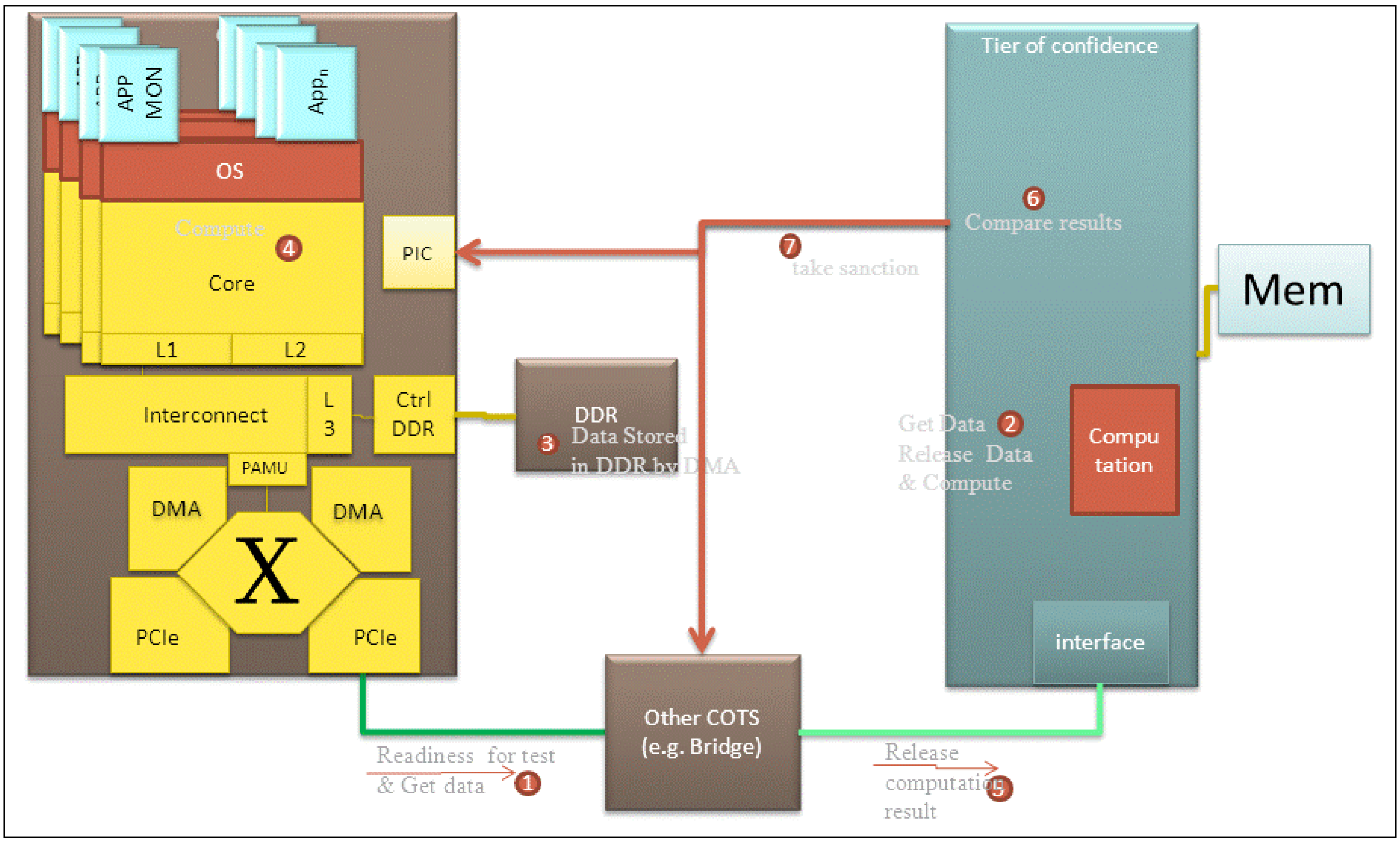


Figure 109: Static view of an example of error detection / mitigation by an independent item on the data path.

Note: the redundancy with a voter is a mitigation mechanism different from the monitoring described here. In such mechanism the applicative computation performed by the COTS of interest (primary computation channel) is performed in a simplified way in a secondary computation channel, dissimilar from the principal. This second channel can be ensured by a COTS microcontroller but in general simpler from the first one. A third actor checks the plausibility of the primary channel computation and takes the appropriate sanctions. This mechanism is performed not on sampled data but in the computation flow.

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

9.3.4.4.3 Summary of failure modes covered

The status of such a mechanism on the basic failure modes use in this report is detailed in Table 27.

Failure mode	Sub class of failure mode	Mitigation argument
Loss of message		In case of loss of message by one of the COTS or COTS blocks on the data path, a simple monitoring of incoming frame reception can detect this mode.
Untimely transfer of message	Babbling Delayed message	Independent item should receive limited number of messages from the MON partition during the time slice. In this time slice multiple receptions can track a babbling and delayed reception can be monitored.
Abnormal sequence of message		In the basic mechanism, too few messages are received from the MCU to check an abnormal sequence. More elaborated version may include it but necessitate more exchanges.
Untimely or forbidden transition	Untimely transition of data Untimely transition of address (emitter or receiver) Forbidden transition of address (emitter or receiver)	The mechanism is dedicated to the research of such failure. • If data is modified, the result of comparison will be wrong. • If address is modified and the communication is done via a network the result will never reach the Independent item and result will be considered as lost. • If address is modified and the communication is done via a bus the result will never reached the correct address zone of the independent item and result will be considered wrong.
Impossible transition	Impossible transition of data Impossible transition of addresses	As test data are refreshed, impossible transition of data will be systematically detected. In the basic mechanism, with only one test pattern, an address stuck could not be detected. However In case of communication via a bus it is possible to refine this simple mechanism, considering that the test result is sent alternatively to two different zones of the independent item memory.

Table 27: coverage of detection / mitigation loop

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

9.3.4.5 Watchdog and time monitoring

Watchdog can be implemented either as COTS features or as architectural means. We will concentrate here on architectural watchdog[60]. In this case detection is performed by the watchdog that is not refreshed after the correct period and it realises also the mitigation. Architectural watchdogs are various and more or less sophisticated[61]. Two typical examples are presented hereafter.

9.3.4.5.1 Basic watchdog

Principle
A basic watchdog is a mechanism refreshed periodically by the device controlled. If the watchdog is not refreshed after a maximal delay, the watchdog enforces a safe state or procedure for instance a reset order to the device.

Note: the term "device" is employed because use of watchdog is more general than MCU.

This mechanism is in particular dedicated to cover global failures. Avionics watchdog have an independent clock from the device clock.,

COTS-AEH_Suggestion_15.	<u>Monitoring by a Watchdog</u> During COTS design error mitigation strategy elaboration, if a monitoring by a watchdog is defined in order to detect COTS design error, o Independence between the watchdog and the monitored device should be assessed (e.g. independent clock reference).

Failure covered
This mechanism is dedicated to cover *Loss of messages* and *Untimely transfer of messages (delayed)* considering that a device that cannot refresh its watchdog cannot correctly function.
A basic watchdog with refresh period configured as closed as possible of the time frame could detect a real time drift.

9.3.4.5.2 Time Windows-based watchdog

Principle
A Time Windows based watchdog is a mechanism refreshed periodically by an operating system or by applicative software, at some program step. If the watchdog is not refreshed in a precise time-slice, the watchdog detects misbehaviour and enforces a safe state or procedure for instance a reset order to the software or to the hardware.

Failure covered
This mechanism is dedicated to cover *Loss of messages* and *Untimely transfer of messages (delayed and advanced)* due to time drift.

[60] Subject to obtain data necessary to assess the independence between an internal watchdog and the blocks it monitors, internal watchdogs could become admissible mechanisms
[61] Most sophisticated watchdog can be comparable to the monitoring by a tier of confidence studied in preceding section.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9.3.4.6 Mechanisms for memories

In addition to ECC and parity defined on memories it is possible to add some other mechanisms like CRC, Check sum or data mirroring. These mechanisms are in general implemented at applicative level or by the Operating System. They should be limited to critical data or blocks of data. Their end-to-end nature guarantees the integrity of the transfer from the memory up-to the core when the ECC guarantees only the storage in the memory and the transfer from memory to the memory controller.
Consider for instance the transfer from an EEPROM to a DRAM and then to the core. A Power-on Built in Test (PBIT) verify that the DRAM is able to host the software, then the ECC is added in the transfer from EEPROM to DRAM in order to guarantee the forthcoming transfer from DRAM to the DRAM Controller. Additional mechanisms can be added in order to guarantee that the DRAM executed program is the same that the program initially loaded in EEPROM (for instance CRC).

9.3.4.6.1 Checksum

Principle
A block of memory containing critical data is processed by the initiator of data storage (in general the processor), prior to the write operation, through a given operation on the data. The result is stored in memory. When the data is read the same computation is performed by the target of data and the result is compared with the initial checksum. In case of detection of discrepancy between both values, the target takes a sanction. Please see [26] part 7 for further extension.

<u>Note</u>: the checksum should be stored as far as possible from memory zone where the data are stored even when possible on another page or memory module.

Failure covered
Checksum covers errors of the complete chain between the encoding block and the decoding block. It participates to an end-to-end mechanism applied to memory.
Failure modes covered are:

- Untimely transfer of message (advanced)
  When generated by the memory controller;
- untimely and forbidden transition of data and addresses;
- Impossible transition of data and addresses.

9.3.4.6.2 Cyclic Redundancy Check

Principle
Cyclic Redundancy Check (CRC) is a bit position dependant checksum. The block of memory containing critical data is processed through a polynomial division by a given polynomial. The rest of this division (CRC signature) is stored in a memory. When the block of data is read the CRC is computed and compared with the stored value. Discrepancy between encoded and decoded CRC detect failures on data or addresses. The decoder entity can then take a sanction and mitigate the error. Please see [26] part 7 for further extension.

Note: the CRC signature should be stored as far as possible from memory zone where the data are stored even when possible on another page or memory module.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

Failure covered

CRC covers errors of the complete chain between the encoding block and the decoding block. It participates to an end-to-end mechanism applied to memory. The interest of this mechanism compared to the ECC is that it covers also the buried blocks of the COTS that initiate the write and/or the read operations.

Failure modes covered are:

- o Untimely transfer of message (advanced)
  When generated by the memory controller;
- o untimely and forbidden transition of data and addresses;
- o Impossible transition of data and addresses.

9.3.4.6.3 Data Mirroring

Principle

When a critical data is written in memory a complement to 0 of this data is written in another zone of memory (another page or another module). When this data is requested the data and its mirror are read in memory. A sum of data and its mirror is performed before to use the data. If the result is 0 the data is correct, if not it is rejected.

This method has the advantages to detect 100% of data corruptions, to localize the error and to lead to simple computation (complement to 0, sum). It consumes more data space than Checksum and CRC so that it is in general use for some individual critical applicative data and not blocks of data.

Failure covered

Failure modes covered are:

- o Untimely transfer of message (advanced)
  When generated by the memory controller;
- o untimely and forbidden transition of data and addresses;
- o Impossible transition of data and addresses.

The following suggestion is applicable to the three preceding mechanisms.

COTS-AEH_Suggestion_16.	<u>Error mitigation of memories and memory controllers</u> During COTS design error mitigation strategy elaboration, if a memory monitoring based on information addendum (Checksum, CRC, data mirroring) is defined, o The information added should be segregated as much as possible from the protected data.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

9.3.5 Monitoring Mechanisms for latent failures

9.3.5.1 General principle

Mechanisms described in the preceding sections are dedicated to detection and mitigation of failures caused by design errors. These mechanisms can themselves be affected by different type of failures:

- Systematic design errors or specification failures,
- Random hardware failures, radiation initiated failures (e.g. SEU) or manufacturing systematic failures.

On first side, systematic design and specification failures of detection / mitigation mechanisms should be revealed during fault injection tests (see subsection 9.2.3.4).

On second side, periodic monitoring should be defined in order to guarantee that detection and mitigations means are not inoperative due to some failure (typically random failures). This monitoring is typically performed at Power-On.

COTS-AEH_Suggestion_17.	Avoidance of latent failure of detection / mitigation mechanisms During COTS design error mitigation strategy elaboration, if a detection and/or monitoring mechanism is defined in order to detect COTS design error, ○ this mechanism should be monitored in order to keep latent failures under control; unless it is shown that the defined mechanism ○ is free of design error, ○ is sufficiently reliable.

9.3.5.2 Example: protection of microcontroller internal memories

Microcontroller caches: L1, L2, L3, TLB (MMU cache), PAMU Cache are in general protected by ECC and/or parity [62]. In the same manner and DDR controllers support ECC that is the principal failure mitigation available on DDR.

In order to outline the possibilities of these mechanisms, consider for instance ECC on the L3 Cache (CPC) of P4080.

An error detected on a data stored in CPC will be signalled by some signal in CPC Error Register MULLERR (Multiple CPC ERRors) ([33], 8.3.1.3). If the error is not correctable or if the error is correctable but the error counter has reached its maximum value, a machine check interrupt (directly to the core) is generated for the operating system in order to notice and handle the error.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Attention has to be paid to the fact that error reporting and handling are configurable in the CCSR (CPC ERRor INTerrupt ENable register: CPCERRINTEN).
Single errors are corrected by ECC on the data transmitted. Action to be taken has to be taken by the CPU under dedicated privilege mode: supervisor or hypervisor depending of the processor type. Operating System is in charge to correct the error also in the cache and to restore the error reporting register.

This example can be, in its principle, generalized to other cache and memories.

Freescale microcontrollers propose a particular feature that allows checking ECC or Parity correct behaviour by injecting errors in memories. This operation is crucial to test before each operation the validity and the correct settings of the integrity protection mechanism[62] and is typically implemented in Power-on Built-In-Self Tests (PBIT). It has of course to be operated carefully and adequate measures should be taken in order to forbid its triggering during flight.

In parallel to these mechanisms, in order to reduce the exposure to cache errors, periodic flush of cache can be operated [62]. This mechanism is in general dedicated to cover SEU induced failures.

[62] By integrity protection mechanism we mean a mechanism allowing the detection and mitigation of failure like untimely, forbidden or impossible transition of information.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE TECHNICAL LIBRARY ABBOTTAEROSPACE.COM

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

10 CONCLUSION AND RESULT SUMMARY

This chapter aims at supplying a critical summary of the method, the applicability of its various elements and finally giving some highlights to try to define one or several stop criteria.

Indeed, no stop criteria can be definitely defined: information provided in this document is designed to enlighten the complex COTS mastering concern. This information should then not be considered as obligations for industrials to qualify COTS on equipment.
.

10.1 SUMMARY OF REPORT ACTIVITIES

At this stage of the study, it is necessary to bring a critical viewpoint on the process followed and the results obtained. The preliminary step of breakdown and abstraction levels will be broached then the general method will be summarized.

10.1.1 Breakdown and Abstraction levels

The present study is firstly based on the choice of some breakdown and abstraction levels. This choice is crucial and provides the framework for the complete study.
It has been chosen to work both at black box level and at grey box level. Grey box is a black box refined breakdown level, built with fragmentary, non-contractual and potentially under NDA information.

- Black box level allows identifying external media in relations (e.g. bus and networks), the interfaces type and having a clear vision of the functions of the COTS – its responsibilities with respect to its context.
- Grey box allows having some insights in its constitution and intrinsic failure modes.

At both breakdown levels, logical abstraction level appears to be the best compromise between generality of functional level and the too great details of physical level. Nevertheless, functional level has been used for determination of COTS functions at black box level and of blocks functions at grey box level. Inroads at physical level have been conducted on the basis of logical level failure modes for refinement of some failure mechanisms (e.g. TTL, clock failure modes, etc.).

10.1.2 Global analysis process

The process deployed in this report fit into this framework and relies on different patterns and methods:

- The choice of a failure model at Logical level;
- A preliminary black box description focusing on COTS output with
    - Description of considered outputs;
    - Analysis of COTS output failure modes;
    - List of I/O integrated detection/ mitigation mechanisms;

    This approach provides a view on COTS internal failure mode from the point of view of one output failure mode.

- A second approach at grey box level with

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

- Choice of an architecture;
- Description of considered blocks;
- Analysis of COTS block failure modes;
- List of internal detection/ mitigation mechanisms;

  This approach provides a view on COTS internal failures that can impact various outputs. Grey box approach appears important in order to understand as finely as possible the internal behaviour of the COTS, its logical structure (block breakdown). Its physical structure remains in general non accessible. The analysis of various families of COTS suggests that some few patterns could generate most of the systematic failures.

- The use of tests and the determination of mitigation means, split in three categories:
  - Mechanisms relying on internal mitigation means of the COTS;
  - Mixed mechanisms relying on internal COTS detection means and on external mitigation means;
  - Mechanisms fully relying on architecture means for detection and mitigation63.

Failure model is a crucial pattern use in the study, its comprehensiveness is crucial because a missing failure mode may cause the invalidation of the analysis at both black box and grey box levels and the judgement on the detections mechanism coverage. At grey box level, the selection of COTS internal architecture appears also to be a tactical choice for the relevancy of failure analysis.

10.2 IMPLEMENTATION STRATEGY PROPOSAL

Definition of a stop criterion in this framework could be:

What are the necessary elements of proof to be brought in order to ensure that a COTS usage does not present unacceptable risk in an avionic context?

The present report is not intended to bring a solution to this problem, it allows however to draw some tracks[64].

Two axes are suggested by the previous formulation:

- the reliability, when it is possible to guarantee that no error can occur and
- the mitigation of the errors when that is not possible.

It seems clear in the progress of the report that a COTS is an aggregate of blocks or IP (Intellectual Properties) in interaction. The complexity of the COTS arises from the complexity of the interaction of these IP (let us consider for example the interactions of a PCIe block with a DMA) and with the complexity

[63] These detection/ mitigation techniques are varied and only a representative panel is depicted here. For this reason only recommendations and good practices were provided within the report and no suggestions for complementary or amendments to EASA guidance as requested in subchapter 2.2.

[64] Indeed, for random failures, the stop criterion is provided by probabilistic targets [1], for in-house developed component the stop criterion is provided by a development effort linked to the Design Assurance Level [4]. However, no standardize criterion exist in order to determine if the study of a COTS design is sufficient or not for a given DAL.

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

of these IP themselves (let us consider for example an interconnect block). This fundamental complexity is amplified by an impossibility to know the details of the functioning of some of these IP and their detailed interactions.

Black box tests in general and endurance tests in particular allow going through a large number of interactions on the COTS and thus through particular activation of the IP and their interactions.

These tests turn out to be powerful tools to demonstrate the reliability of some IP. It is in particular the case for the simple IP in interface with COTS environment (let us consider for example SPI controller) and maybe in the case of complex IP in simple interaction with other blocks.

For the IP, which complexity or interactions, do not allow to guarantee the reliability by the test, a grey box study turns out to be necessary. The type of studies developed in chapter 8 allows defining the failures of the considered IP and their impact on the COTS outputs.

Some of these errors can arise from tests, from errata, or being considered as possible considering the accessible elements of the IP design.

Mechanisms such as those proposed or alternative ones allow protection against these failures (primary mechanisms). These primary mechanisms detect and mitigate failures that could directly impact safety.

Primary mechanisms should themselves be tested during development (fault injection tests) and be periodically monitored in operation in order to avoid latent failures.

Secondary mechanisms should be defined to prevent primary mechanisms from adverse deactivation caused by systematic error or random fault.

This Document Provided by the Abbott Aerospace Technical Library
abbottaerospace.com

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

11 REFERENCES

[1] EUROCAE & SAE, ED-79A/ARP-4754A: Guidelines for Development of Civil Aircraft and Systems, European Organisation for Civil Aviation Equipment (EuroCAE) & Society of Automotive Engineers (SAE), 2010.

[2] EUROCAE & RTCA, ED-12B / DO-178B: Software considerations in airborne systems and equipment certification, 1992.

[3] EUROCAE & RTCA, ED-12C/DO-178C: Software Considerations in Airborne Systems and Equipment Certification, January 2012.

[4] EUROCAE & RTCA, ED-80/DO-254:Design Assurance Guidance for Airborne Electronic Hardware, April 2000.

[5] EASA, "EASA CM - SWCEH – 001 Iss. 1 Rev. 1: Development Assurance of Airborne Electronic Hardware,," 9th Mar. 2012.

[6] F. Faubladier et D. Rambaud, «EASA.2008/1, 2008: SoC Survey Report - Safety Implications of the use of system-on-chip (SoC) on commercial of-the-shelf (COTS) devices in airborne critical applications,» EASA – Research Project., 2008.

[7] RTCA, DO-297 - Integrated Modular Avionics (IMA) development, guidance and certification consideration, Radio Technical Commission for Aeronautics (RTCA), 2005.

[8] «Wikipedia,» [En ligne]. Available: www.wikipedia.org.

[9] International Standard Organisation - WG16, «ISO 26262: RoadVehicle Functional Safety,» International Standard Organisation , 2011.

[10] H. Forsberg et K. Karlsson, «COTS CPU Selection Guidelines for Safety-Critical Applications,» chez *25th Digital Avionics Systems Conference, IEEE/AIAA, 1-12,* http://dx.doi.org/10.1109/DASC.2006.313701, 2006.

[11] B. Green, J. Marotta, B. Petre, K. Lillestolen, R. Spencer, N. Gupta, D. O’Leary, J. Lee, J. Strasburger, A. Nordsieck, B. Manners et R. and Mahapatra, «DOT/FAA/AR-11/2: Handbook for the Selection and Evaluation of Microprocessors for Airborne Systems,» Federal Aviation Administration, February 2011.

[12] R. Mahapatra et S. Ahmad, «DOT/FAA/AR-06/34: Microprocessor evaluations for safety-critical, real-time applications: authority for expenditure no. 43 phase 1 report,» FAA, December 2006.

[13] R. Mahapatra, P. Bhojwani et J. Lee, «DOT/FAA/AR-08/14: Microprocessor evaluations for safety-critical, real-time applications: authority for expenditure no. 43 phase 2 report,» FAA, June 2008.

[14] R. Mahapatra, P. Bhojwani, J. Lee et Y. Kim, «DOT/FAA/AR-08/55: Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 3 Report,» FAA, February 2009.

[15] R. Mahapatra, J. Lee, N. Gupta et B. Manners, «DOT/FAA/AR-10/21: Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 4 Report,» FAA, September 2010.

[16] R. Mahapatra, J. Lee, N. Gupta et B. Manners, «DOT/FAA/AR-11/5: Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 5 Report,» FAA, May 2011.

[17] X. Jean, M. Gatti, G. Berthon et M. Fumey, «EASA.2011/6 - The Use of Multicore processors in

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

Airborne Systems (MULCOR),» EASA, November 2012.

[18] International Standard Organization / Internation Electrotechnical Commission, “ISO/IEC 7498-1 Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model”.

[19] Department of Defense, «MIL-STD-1553B Notice 4 : Interface Standard for Digital Time Division Command/Response Multiplex Data bus,» Department Of Defense - United State of America, 1996.

[20] Data Device Corporation, «Military Flight Control Solutions for Cost-Effective Commercial Avionics,» Data Device Corporation, [En ligne]. Available: http://www.ddc-web.com/commercial/default.htm. [Accès le 11 09 2013].

[21] Data Device Corporation, “PCI-Express AceXtreme, Product Brief DD-42900 Rev 1,” Data Device Corporation, Bohemia (New-York), 2013.

[22] Micron Technology, Inc., “GENERAL DDR SDRAM FUNCTIONALITY - TN-46-05,” *Micron DesignLine*, vol. 8, no. 3, pp. 65-76, 2001.

[23] F. MASUOKA, M. MOMODOMI, Y. IWATA et R. SHIROTA, «New ultra high density EPROM and flash EEPROM with NAND structure cell,» chez *Electron Devices Meeting, 1987 International (Volume:33)* , 1987.

[24] K. Faxén, C. Bengtsson, M. Brorsson, H. Grahn, E. Hagersten, B. Jonsson, C. Kessler, B. Lisper, S. P. et B. Svensson, «Multicore computing—the state of the art,» chez *Multicore Days organized by SICS, Swedish Multicore Initiative and Ericsson Software Research*, 2008.

[25] V. Brindejonc, G. Marcuccilli et S. Petit, «System FMECA in the framework of ISO 26262,» chez *Lambda-Mu 17*, La Rochelle, October 2010.

[26] International Electrotechnical Commission, IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems - Ed 2.0, International Electrotechnical Commission, 2010.

[27] M. Squair, “Safety, Software Architecture and MIL-STD-1760.,” in *Proc. Eleventh Australian Workshop on Safety-Related Programmable Systems (SCS 2006)*, Melbourne (Australia), 2006.

[28] Department of Defense, «MIL-STD-1760D: Interface Standard for Aircraft-store Interconnection System (AEIS),» Department Of Defense - United state of America, 2003.

[29] PCI-Special Interest Group, “PCI Local Bus Specification, Revision 2.2,,” PCI Special interest group, December 18, 1998,.

[30] S. Beaulieu, «Analyse du déterminisme et de la fiabilité du protocole PCI express dans un contexte de certification avionique,» ÉCOLE DE TECHNOLOGIE SUPÉRIEURE - UNIVERSITÉ DU QUÉBEC, 2012.

[31] A. Asseo, “PCI Express Protocol Studies and Characterization For Integration in The Avionics Context,” Thales, 2008.

[32] PCI-Special Interest Group, “PCI Express Base Specification 2.0,” PCI-SIG, 2006.

[33] Freescale Semiconductor, Inc., “P4080 QorIQ Integrated Multicore Communication Processor Family Reference Manual - Document Number: P4080RM Rev. 1,,” Freescale Semiconductor - Refence Manual, Austin (USA), January 2012.

[34] Integrated Device Technology, Inc., “IDT Tsi384 PCIe-to-PCI Bridge User Manual ; 80E1000_MA001_10,” Integrated Device Technology, Inc., San Jose, California 9513, 2009.

[35] Integrated Device Technology, Inc., «IDT Tsi384 Device Errata, 80E1000_ER001_09,» Integrated Device Technology, Inc., San Jose, California 95138 , 2009.

[36] J. Ajanovic, Jackson and Carl, “PCI Express™ Advanced Hardware Topics & Specification Updates,”

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

2004. [Online]. Available: http://www.pcisig.com/developers/main/training_materials/#2004. [Accessed 06 09 2013].

[37] Holt Integrated Circuit, Inc., "HI-3585, HI-3586 : ARINC 429 Terminal IC with SPI Interface - (DS3585 Rev. I)," HOLT INTEGRATED CIRCUITS, 2012.

[38] Data Device Corporation , "DD-00429: ARINC 429 MICROPROCESSOR INTERFACE - Datasheet rev L," Data Device Corporation , Bohemia (New-York), 2010.

[39] Data Device Corporation, "DD-42900: ARINC 429 MICROPROCESSOR INTERFACE DEVICE, Datasheet," Data Device Corporation, Bohemia (New-York), 1999.

[40] AIRLINES ELECTRONIC ENGINEERING COMMITTEE, "ARINC Specification 429 Part 1-17: Mark 33 – Digital Information Transfer System (DITS)," AERONAUTICAL RADIO, INC, Annapolis (MA), 2004.

[41] Data Device Corporation, "PCI-Express AceXtreme® - Datasheet , Rev A," Data Device Corporation, Bohemia (New-York), 2013.

[42] Micron Technology, Inc., «1Gb: x4, x8, x16 DDR3 SDRAM Features,» Micron Technology, Inc., 2006.

[43] JEDEC, «DDR3 SDRAM Standard,» JEDEC Solid State Technology Association, Arlington (USA), 2012.

[44] GreenLiant, "NAND Controller GLS55VD031 - Datasheet S71397-04-000," Greenliant Systems, Ltd., 2010.

[45] GreenLiant, "NAND Controller GLS55VD020 - Datasheet S71355-04-000," Greenliant Systems, Ltd., 2010.

[46] Lattice Semiconductor, Inc., "NAND Flash Controller - RD1055 v01.2," Lattice Semiconductor, Inc., 2010.

[47] QuickLogic, "NAND Flash controller Datasheet Rev A," QuickLogic Corporation, 2008.

[48] Spansion, "S34ML08G1 NAND Flash Memory for Embedded Data Sheet (Advance Information) Release 05," Spansion, 2013.

[49] Spansion, «Hyperstone F2 and F3 NAND Controllers/Spansion® NAND – Enhancing Power Fault Tolerance - Application Note,» Spansion Inc, 2013.

[50] Freescale Semiconductor, Inc., «MPC8610 Integrated host processor Fact sheet; MPC8610FS Rev. 2,» Freescale Semiconductor - Fact Sheet, Austin (USA), 2007.

[51] Freescale Semiconductor, Inc., «MPC8610 Integrated Host Processor Hardware Specifications - MPC8610EC - Rev 2.,» Freescale Semiconductor - Datasheet, Austin (USA), 2009.

[52] Freescale Semiconductor, Inc., "MPC8610 Integrated Host Processor Reference Manual - MPC8610RM, Rev. 1.," Freescale Semiconductor - Reference Manual, Austin (USA), 2010.

[53] Freescale Semiconductor, Inc, "e600 PowerPC™ Core Reference Manual - E600CORERM - Rev. 0,," Freescale Semiconductor - Reference Manuals, Austin (USA), 2006.

[54] Freescale Semiconductor, Inc., "MPC8610 Chip Errata - MPC8610CE - Rev 0.," Freescale Semiconductor - Chip Errata, Austin (USA), 2010.

[55] N. Garinger, M. Dorr, M. Naumann and G. Walker, "On Chip Network". US Patent US Patent 7.277.449B2, 2 October 2007.

[56] Freescale Semiconductor, Inc., "Outstanding Data Tenures on the MPX Bus - AN2161 - Rev 0.1," Freescale Semiconductor, Inc., Austin (USA), 2004.

[57] Freescale Semiconductor, Inc., «QorIQ™ P4080 Communications Processor Product Brief -

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

Document Number: P4080PB Rev. 1,» Freescale Semiconductor- Product Brief, Austin (USA), September 2008.

[58] Freescale Semiconductor, Inc., "P4080/P4081 QorIQ Integrated Processor Hardware Specifications," Freescale Semiconductor - Data Sheet: Technical Data, Austin (USA), April 2013.

[59] Freescale Semiconductor, "e500mc Core Reference Manual, Rev 3.," Freescale Semiconductor, 2013.

[60] Freescale Semiconductor, «EREF 2.0: A Programmer's Reference Manual for Freescale Power Architecture® Processors - Rev. 0,» Freescale Semiconductor Reference Manual, September 2011.

[61] Freescale Semiconductor, Inc., "P4080 Chip Errata, P4080CE Rev. N," Freescale Semiconductor - Chip Errata, Austin (USA), 2012.

[62] P. Genua, «Error Correction and Error Handling on PowerQUICC™ III Processors - Document Number: AN3532 Rev. 0,» Freescale Semiconductor Application Note, November 2007.

[63] E. Bost, «Hardware Support for Robust Partitioning in Freescale QorIQ Multicore SoCs (P4080 and derivatives),» Freescale Semiconductor White Paper - QORIQHSRPWP - Rev. 0, May 2013.

[64] Z. Gu et Q. Zhao, «A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization,» *Journal of Software Engineering and Applications,* vol. 05, pp. 277 - 291, 2012.

[65] X. Jean, D. Faura, M. Gatti, L. Pautet et T. Robert, «Software Approach for Managing Shared Resources in Multicore IMA Systems,» chez *Digital Avionics Systems Conference (DASC),* Syracuse (USA), 2013 ,IEEE/AIAA 32st.

[66] S. Deshpande, "Flow Control Mechanisms for Avoidance or Retries and/or Deadlocks in an Interconnect". United-State Patent US 2010/0318713 A1, 16 12 2010.

[67] Freescale Semiconductor, Inc;, "P5020 Chip Errata, P5020CE rev. J," Freescale Semiconductor - Chip errata, Austin (USA), 2013.

[68] PCI-Special Interest Group, "PCI Express® Compliance Testing," PCI-Special Interest Group, 2013. [Online]. Available: http://www.pcisig.com/specifications/pciexpress/compliance. [Accessed 15 09 2013].

[69] P. Dervin, "Method of testing data paths in an electronic circuit". USA Patent US 7913129 B2, 22 03 2011.

[70] V. Brindejonc and N. Plaze, "Rédaction, vérification et gestion des exigences de Sûreté de Fonctionnement," in *Lambda-Mu 18,* Tours, 2012.

[71] Freescale Semiconductor, Inc, "P5020 QorIQ Integrated Multicore Communication Processor Family Reference Manual - P5020RM, Rev 3.," Freescale Semiconductor, Inc, Austin (Tx), 2013.

[72] C. E. Shannon, «A Mathematical Theory of Communication,» *The Bell System Technical Journal,* vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[73] J. Cook, "Flash memory 101: An Introduction to NAND flash," EE-Times - Connecting the Global Electronics Community, 20 March 2006. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1272118. [Accessed 13 September 2013].

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

ANNEX 1: MAPPING OF A MULTICORE CONFIGURATION, CONTROL AND STATUS REGISTER (CCSR)

This table is reworked on the basis of data present in [71].

Block Base Address (Hex)	Block	Section/ Page	Comments	volume of the zone
0x00_0000 - 0x00_0FFF	Local access control-Local configuration control Local access control-Local access windows	Local Configuration Control Memory Map	-	4095
0x00_1000 - 0x00_7FFF	Reserved	-	-	28671
0x00_8000 - 0x00_8FFF	DDR memory controller 1	DDR Memory Controller Memory Map	-	4095
0x00_9000 - 0x00_9FFF	DDR memory controller 2	DDR Memory Controller Memory Map	P5020 only; not present on P5010	4095
0x00_A000 - 0x00_FFFF	Reserved	-	-	24575
0x01_0000 - 0x01_0FFF	CoreNet TM platform cache 1 (CPC1)	CoreNet TM Platform Cache (CPC) Memory Map	-	4095
0x01_1000-0x01_1FFF	CoreNet TM platform cache 2 (CPC2)	CoreNet TM Platform Cache (CPC) Memory Map	P5020 only; not present on P5010	4095
0x01_2000-0x01_7FFF	Reserved	-	-	24575
0x01_8000-0x01_8FFF	CoreNet TM coherency fabric (CCF)	CoreNet TM Coherency Fabric (CCF) Memory Map	-	4095
0x01_9000-0x01_FFFF	Reserved	-	-	28671
0x02_0000-0x02_0FFF	PAMU partition 1	PAMU Memory Map	The PAMU is partitioned into 16 identical instances. Not all	4095

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Block Base Address (Hex)	Block	Section/ Page	Comments	volume of the zone
0x02_1000- 0x02_1FFF	PAMU partition 2	PAMU Memory Map	are necessarily backed with physical hardware. However, all of them must be programmed identically or undefined behaviour may result.	4095
0x02_2000- 0x02_2FFF	PAMU partition 3	PAMU Memory Map		4095
0x02_3000- 0x02_3FFF	PAMU partition 4	PAMU Memory Map		4095
0x02_4000- 0x02_4FFF	PAMU partition 5	PAMU Memory Map		4095
0x02_5000- 0x02_5FFF	PAMU partition 6	PAMU Memory Map		4095
0x02_6000- 0x02_6FFF	PAMU partition 7	PAMU Memory Map		4095
0x02_7000- 0x02_7FFF	PAMU partition 8	PAMU Memory Map		4095
0x02_8000- 0x02_8FFF	PAMU partition 9	PAMU Memory Map		4095
0x02_9000- 0x02_9FFF	PAMU partition 10	PAMU Memory Map		4095
0x02_A000- 0x02_AFFF	PAMU partition 11	PAMU Memory Map		4095
0x02_B000- 0x02_BFFF	PAMU partition 12	PAMU Memory Map		4095
0x02_C000- 0x02_CFFF	PAMU partition 13	PAMU Memory Map		4095
0x02_D000- 0x02_DFFF	PAMU partition 14	PAMU Memory Map		4095
0x02_E000- 0x02_EFFF	PAMU partition 15	PAMU Memory Map		4095
0x02_F000- 0x02_FFFF	PAMU partition 16	PAMU Memory Map		4095
0x03_0000- 0x03_FFFF	Reserved	-	-	65535
0x04_0000- 0x04_FFFF	MPIC-Global registers	MPIC Memory Map	Global configuration: 0x04_1000 Global timers: 0x04_1100	65535
0x05_0000- 0x05_FFFF	MPIC-Interrupt source registers	MPIC Memory Map	External IRQs: 0x05_0000 Internal IRQs: 0x05_1200	65535
0x06_0000- 0x06_FFFF	MPIC-Processor (core) registers	MPIC Memory Map	-	65535

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Block Base Address (Hex)	Block	Section/ Page	Comments	volume of the zone
0x07_0000-0x0B_FFFF	Reserved	-	-	327679
0x0C_0000-0x0C_FFFF	RapidIO Architectural registers	SRIO Memory Map/Register Definition		65535
0x0D_0000-0x0D_FFFF	RapidIO Implementation registers	SRIO Memory Map/Register Definition		65535
0x0E_0000-0x0E_0FFF	Configuration/pin control	Device Configuration and Pin Control Memory Map/Register Definition		4095
0x0E_1000-0x0E_1FFF	Clocking	Clocking Memory Map/ Register Definition		4095
0x0E_2000-0x0E_2FFF	Run control/power management (RCPM)	RCPM Memory Map/ Register Definition		4095
0x0E_3000-0x0E_7FFF	Reserved	-	-	20479
0x0E_8000-0x0E_8FFF	Security fuse processor (SFP)	Security fuse processor (SFP) memory map		4095
0x0E_9000-0x0E_9FFF	Reserved	-	-	4095
0x0E_A000-0x0E_AFFF	SerDes control	SRDS Memory Map/ Register Definition		4095
0x0E_B000-0x0F_FFFF	Reserved	-	-	86015
0x10_0000-0x10_0FFF	DMA controller 1	DMA controller memory map		4095
0x10_1000-0x10_1FFF	DMA controller 2	DMA controller memory map		4095
0x10_2000-0x10_FFFF	Reserved	-	-	57343
0x11_0000-0x11_0FFF	Enhanced serial peripheral interface (eSPI)	Enhanced serial peripheral interface (eSPI) memory map		4095
0x11_1000-0x11_3FFF	Reserved	-	-	12287
0x11_4000-0x11_4FFF	Enhanced secure digital high capacity (eSDHC)	eSDHC memory map/ register definition		4095
0x11_5000-0x11_7FFF	Reserved	-	-	12287

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

Block Base Address (Hex)	Block	Section/ Page	Comments	volume of the zone
0x11_8000-0x11_8FFF	Dual I²C controller 1	I²C Controller Memory Map	I²C 1: 0x11_8000 I²C 2: 0x11_8100	4095
0x11_9000-0x11_9FFF	Dual I²C controller 2	I²C Controller Memory Map	I²C 3: 0x11_9000 I²C 4: 0x11_9100	4095
0x11_A000-0x11_BFFF	Reserved	-	-	8191
0x11_C000-0x11_CFFF	DUART controller 1	DUART Memory Map/ Register Definition	UART1: 0x11_C500 (DUART1) UART2: 0x11_C600 (DUART1)	4095
0x11_D000-0x11_DFFF	DUART controller 2	DUART Memory Map/ Register Definition	UART3: 0x11_D500 (DUART2) UART4: 0x11_D600 (DUART2)	4095
0x11_E000-0x12_3FFF	Reserved	-		24575
0x12_4000-0x12_4FFF	Enhanced local bus controller (eLBC)	Enhanced Local Bus Controller (eLBC) Memory Map		4095
0x12_5000-0x12_FFFF	Reserved	-		45055
0x13_0000-0x13_0FFF	GPIO controller	GPIO Memory Map/Register Definition		4095
0x13_1000-0x13_7FFF	Reserved	-		28671
0x13_8000-0x13_8FFF	Pre-boot loader (PBL)	Reserved Address Space Used as Internal PBL Commands	Software cannot write to the PBL CCSR space directly. However, special PBL commands may be leveraged during pre-boot initialization by referencing specific CCSR offsets (unique commands have unique CCSR offsets). See Reserved Address Space Used as Internal PBL Commands," for more information.	4095
0x13_9000-0x1D_FFFF	Reserved			684031
0x1E_0000-0x1E_3FFF	RMan	See"QoRIQ Datapath Acceleration Architecture Reference Manual"		16383
0x1E_4000-0x1F_FFFF	Reserved			114687

This Document Provided by the Abbott Aerospace Technical Library abbottaerospace.com

Block Base Address (Hex)	Block	Section/ Page	Comments	volume of the zone
0x20_0000-0x20_0FFF	PCI Express controller 1	PCI Express memory mapped registers		4095
0x20_1000-0x20_1FFF	PCI Express controller 2	PCI Express memory mapped registers		4095
0x20_2000-0x20_2FFF	PCI Express controller 3	PCI Express memory mapped registers		4095
0x20_3000-0x20_3FFF	PCI Express controller 4	PCI Express memory mapped registers		4095
0x20_4000-0x20_FFFF	Reserved			49151
0x21_0000-0x21_0FFF	USB 1 (host only)	USB Memory Map		4095
0x21_1000-0x21_1FFF	USB 2 (dual role)	USB Memory Map		4095
0x21_2000-0x21_3FFF	Reserved			8191
0x21_4000-0x21_4FFF	USB PHY			4095
0x21_5000-0x21_FFFF	Reserved			45055
0x22_0000-0x22_0FFF	SATA 1			4095
0x22_1000-0x22_1FFF	SATA 2			4095
0x22_2000-0x2F_FFFF	Reserved			909311
0x30_0000-0x30_FFFF	SEC 4.2			65535
0x31_0000-0x31_3FFF	Reserved			16383
0x31_4000-0x31_4FFF	Security monitor	Security monitor memory map/ register definition		4095
0x31_5000-0x31_5FFF	Reserved			4095
0x31_6000-0x31_6FFF	Pattern match engine (PME)			4095
0x31_7000-0x31_7FFF	Reserved			4095
0x31_8000-0x31_8FFF	Queue manager (QMan)			4095
0x31_9000-0x31_9FFF	Reserved			4095
0x31_A000-0x31_AFFF	Buffer manager (BMan)			4095

This Document Provided by the Abbott Aerospace Technical Library ABBOTTAEROSPACE.COM

Block Base Address (Hex)	Block	Section/ Page	Comments	volume of the zone
0x31_B000-0x31_FFFF	Reserved			20479
0x32_0000-0x32_FFFF	RAID Engine (RE)			65535
0x33_0000-0x3F_FFFF	Reserved			851967
0x40_0000-0x4F_FFFF	Frame manager			1048575
0x50_0000-0xFF_FFFF	Reserved			11534335
	EOF			

Table 28: Address mapping of P5020 CCSR

This Document Provided by the Abbott Aerospace
Technical Library
abbottaerospace.com

ANNEX 2: SUMMARY OF SUGGESTIONS

This annex collects the suggestions defined in chapter 9.

Application of these suggestions cannot be requested. Nevertheless if a suggestion is applied, the activities proposed should be realized.

Depending upon the preconditions, two types of suggestions are proposed:

- Methodological suggestions: in this case the preconditions are the study phase (mitigation strategy, test, integration);
- Technique suggestion (one occurrence) where precondition is the "in operation" phase.

COTS-AEH_Suggestion_1.	Use of internal detection / mitigation mechanisms In global mitigation strategy definition, the internal mechanisms selected should be: - Specified; - Managed: activation mode, configuration, error handling; - Tested.

COTS-AEH_Suggestion_2.	Verification of information transmitted during tests During test sequences, data transmitted through the COTS should be monitored with respect to the possible failure mode identified.

COTS-AEH_Suggestion_3.	Verification of detection / mitigation mechanisms status during tests During test sequences, COTS internal detection/Mitigation Mechanisms that are embedded in COTS or in COTS interfaces should be monitored and their status should be reported

COTS-AEH_Suggestion_4.	Status verification of inhibited functions during tests During test sequences, both configuration status and outputs of COTS inhibited features should be monitored.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE TECHNICAL LIBRARY ABBOTTAEROSPACE.COM

COTS-AEH_Suggestion_5.	<u>Configuration management of COTS under test</u> When test are performed on several instances of the same COTS: (a) their configuration should be identical; (b) This configuration should be managed in configuration; (c) An impact analysis should be performed in case of modification of configuration during project time (after exploitable tests begun).

COTS-AEH_Suggestion_6.	<u>Verification of Detection and Mitigation Mechanisms</u> During integration tests at various levels, detection / mitigation mechanisms should be tested in particular through fault injection tests.

COTS-AEH_Suggestion_7.	Integration level for definition of detection / mitigation mechanisms During definition of detection / mitigation strategy, COTS design error detection / mitigation mechanisms should be defined as closer as possible from the COTS. System and aircraft levels may be considered only when no detection / mitigation could be implemented locally.

COTS-AEH_Suggestion_8.	<u>Usage of COTS internal detection/mitigation mechanisms</u> During COTS design error mitigation strategy elaboration, a detection / mitigation mechanism implemented within the COTS can be selected if o Its triggering can be monitored during COTS functional and endurance tests; o It can be tested by fault injections tests on COTS; o A mechanism can be implemented in operation in order to cover its latent failures.

COTS-AEH_Suggestion_9.	<u>Usage of COTS internal detection mechanisms</u> During COTS design error mitigation strategy elaboration, a detection / mechanism implemented within the COTS (such as PIC or JTAG blocks) can be selected if : o Its triggering can be monitored during COTS functional and endurance tests; o It can be tested by fault injections tests on COTS; o A mechanism can be implemented in operation in order to cover its latent failures. In this case, mitigations should be applied by an external device.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

COTS-AEH_Suggestion_10.	<u>Usage of COTS output monitoring</u> During COTS design error mitigation strategy elaboration, if a detection mechanism based on COTS outputs monitoring is used o The detection principle should be specified to the message receiver; o The message receiver monitoring implementation should be tested in integration tests (*); o A mechanism can be implemented in operation in order to cover detection mechanism latent failures. (*) the test can be done at board, LRU or system level, depending on the receiver.

COTS-AEH_Suggestion_11.	<u>Periodic frame monitoring</u> During COTS design error mitigation strategy elaboration, if a periodic frame monitoring mechanism is used in the framework of COTS output monitoring: o The latency induced by the confirmation time should be considered.

COTS-AEH_Suggestion_12.	COTS configuration monitoring During operation, any change in COTS critical configuration registers should be detected by a periodical monitoring. This monitoring may be tested in order to avoid latent faults except if the default configuration of unused blocks or features is showed to be innocuous.

COTS-AEH_Suggestion_13.	<u>End-to-End protection</u> During COTS design error mitigation strategy elaboration, if an end-to-end protection mechanism is defined in order to detect COTS design error, o It should be encoded in COTS higher layers (applicative or higher Operating system layers)

COTS-AEH_Suggestion_14.	<u>Monitoring by an external independent item on the data path</u> During COTS design error mitigation strategy elaboration, if a monitoring by an external independent monitoring on the data path is defined in order to detect COTS design error, o The data and applied function should be chosen carefully; o The periodicity of the monitoring should be considered according to the fault tolerance time interval of the system.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

COTS-AEH_Suggestion_15.	<u>Monitoring by a Watchdog</u> During COTS design error mitigation strategy elaboration, if a monitoring by a watchdog is defined in order to detect COTS design error, ○ Independence between the watchdog and the monitored device should be assessed (e.g. independent clock reference).

COTS-AEH_Suggestion_16.	<u>Error mitigation of memories and memory controllers</u> During COTS design error mitigation strategy elaboration, if a memory monitoring based on information addendum (Checksum, CRC, data mirroring) is defined, ○ The information added should be segregated as much as possible from the protected data.

COTS-AEH_Suggestion_17.	<u>Avoidance of latent failure of detection / mitigation mechanisms</u> During COTS design error mitigation strategy elaboration, if a detection and/or monitoring mechanism is defined in order to detect COTS design error, ○ this mechanism should be monitored in order to keep latent failures under control; unless it is shown that the defined mechanism ○ is free of design error, ○ is sufficiently reliable.

THIS DOCUMENT PROVIDED BY THE ABBOTT AEROSPACE
TECHNICAL LIBRARY
ABBOTTAEROSPACE.COM

EUROPEAN AVIATION SAFETY AGENCY
AGENCE EUROPÉENNE DE LA SÉCURITÉ AÉRIENNE
EUROPÄISCHE AGENTUR FÜR FLUGSICHERHEIT

Postal address
Postfach 101253
50452 Cologne
Germany

Visiting address
Ottoplatz 1
50679 Cologne
Germany

Tel	+49_221_89990-000
Fax	+49_221_89990-999
Mail	info@easa.europa.eu
Web	easa.europa.eu