EASA
European Aviation Safety Agency

Final Report EASA_REP_RESEA_2008_1

Research Project:

Safety Implications of the use of system-on-chip (SoC) on commercial of-the-shelf (COTS) devices in airborne critical applications

An agency of the European Union

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.

SOC SURVEY REPORT

EUROPEAN AVIATION SAFETY AGENCY

Authors:
Frédéric FAUBLADIER
David RAMBAUD

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

µP	Micro-processor
A/D	Analog to Digital Converter
AHB	Advanced High-performance Bus
ALU	Arithmetic Logic Unit
APU	Auxiliary Processor Unit
AQEC	Aerospace Qualified Electronic Components
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
ATMU	Address Translation and Mapping Unit
BHT	Branch History Table
CAN	Controller Area Network
CAT	CATastrophic
CEH	Complex Electronic Hardware
CMA	Common Mode Analysis
COTS	Commercial Off-The-Shelf
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CS	Certification Specification
DDR	Double Data Rate
DER	Designated Engineering Representatives
DFS	Dynamic Frequency Switching
DMA	Direct Memory Access
DMSoC	Digital Media System-on-Chip
DPM	Dynamic Power management
DSP	Digital Signal Processor
DUART	Dual Universal Asynchronous Receiver Transmitter
EASA	European Aviation Safety Agency
ECC	Error Checking and Correcting
EDMA	Enhanced Direct Memory Access
ESA	European Space Agency
FAA	Federal Aviation Administration
FAQ	Frequently Asked Question
FAR	Federal Aviation Regulations

FFP	Functional Failure Path
FFPA	Functional Failure Path Analysis
FIFO	First In First Out
FMEA	Failure Mode Effect Analysis
FPGA	Field Programmable Grid Array
FPU	Floating Point Unit
FSL	Fast Simplex Link
GAL	General Array Logic
GPIO	General Purpose Input/Output
GPR	Global Purpose Register
GPL	Global Public License
HCI	Hardware Configuration Index
HDL	Hardware Description Language
HPI	Host port interface
HSID	Hardware Software Interface Document
HW	HardWare
I2C	Inter Integrated Circuit
I/O	Input / Output
IP	Intellectual Property
IPI	Inter Processor Interrupt
LAW	Local Access Window
LBO	Last Buy Order
LRM	Line Replaceable Module
LSU	Load Store Unit
MB	MicroBlaze
MBU	Multiple Bit Upset
McBSP	Multichannel Buffered Serial Ports
MCI	Memory Controller Interface
MCM	MPX Coherency Module
MEU	Multiple Event Upset
MMU	Memory Management Unit
MPX	Multi Processor eXtension
N/A	Not Applicable
NDA	Non Disclosure Agreement
NVIC	Nested Vector Interrupt Controller
OCM	On Chip Memory
OPB	On chip Peripheral Bus

OS	Operating System
PAL	Programmable Array Logic
PHAC	Plan for Hardware Aspects of Certification
PIC	Programmable Interrupt Controller
PCI	Peripheral Component Interconnection
PCIe	PCI Express
PCN	Product Change Notification
PLB	Processor Local Bus
PLD	Programmable Logic Device
PSSA	Preliminary System Safety Assessment
PWM	Pulse Width Modulation
QOS	Quality of Services
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Logic
SERDES	SERializator / DESserializator
SEU	Single Event Upset
SIMD	Single Instruction Multiple Data
SoC	System on Chip
SoPC	System on Programmable Chip
SRIO	Serial Rapid Input Output
SRU	Shop Replaceable Unit
SW	SoftWare
TCP	Transmission Control Protocol
TLB	Translation Lookaside Buffer
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
V&V	Validation & Verification
VHDL	Very high speed HDL
WCET	Worst Case Execution Time

Acknowledgements

The authors wish to express their deep appreciation for the EASA assistance who have provided valuable comments and suggestions. The authors would also like to acknowledge their appreciation for the support of System on Chip providers (Freescale, Texas Instrument, Altera, Xilinx, Actel, ARM) who have provided significant contributions to this research.

Finally, the authors would particularly like to thank Thierry PERRIN and Caroline GARCIA for their significant support.

Executive Summary

The intent of this report is to provide findings about safety issues when using System on Chip on aircraft and submitting potential approaches for addressing these safety concerns. The scope of the survey has been defined by the call for tender referenced EASA.2008.OP.04 "Safety Implications of the use of system-on-chip (SoC) on commercial of-the-shelf (COTS) devices in airborne critical applications".

The survey has been performed in 4 phases with the participation of:

- The European Aviation Safety Agency (EASA),
- System on Chip providers.

First of all, SoC devices and SoC providers candidates have been selected according to their weight in the aeronautical market and more generally in the electronic market. Moreover, their technical solutions are representative of what can be found on the market today.

Then, during phase 2, the public data related to each candidate device has been analyzed in order to point out potential safety issues and difficulties to meet Certification Specification requirements. This assessment work has been conducted under a structured methodology in order to ensure that all the SoC candidates have been addressed with the same rigor. The methodology consists in assessing the public data against the most popular design assurance alternative methods (reverse engineering, architectural mitigation technique, service history, electronic management plan process…), completed with an assessment against key attributes of the ED80/DO254, deemed pertinent for the SoCs.

On the basis of the phase 2 conclusions, it was clear that it may be necessary to get the support of SoC providers and/or have access to their confidential data to be able to meet design assurance standards. Thus, the possibility to involve SoC providers in a certification process and more generally to get some support from them during the implementation of a SoC has been assessed during the third phase.

To finish, the survey has concluded with the identification of potential amendments to the current hardware certification practices for the use of SoC.

.

Background

To remain competitive, the aircraft industry requires to design avionics systems with a high level and ever-growing processing rates demands, which can support multiple standardized international protocols. To achieve these requirements, the design techniques based on full custom design approaches are more and more difficult to implement due to technical complexities versus cost and time to market constraints. It is one of the reasons why the avionic suppliers move towards design techniques like System on Chip (SoC), which consists in integrating into a single circuit heterogeneous electronic functions, and generally implemented as pre-qualified hardware blocks.

However, such systems were initially designed for non aeronautical applications and therefore offer little or no visibility on design assurance.

Moreover, they are often implemented through dedicated software tools, for which the visibility on design assurance is also limited. This lack of visibility is clearly an issue regarding the airworthiness certification and may represent a safety risk, if SoCs are used in safety critical applications.

In this context, the EASA has launched a survey on the safety implications of the use of SoC technology in airborne critical applications, and has mandated Aeroconseil to lead this study.

Aims and Objectives

The first objective of the survey is to assess the safety implication of using SoC in safety critical applications. The assessment, based on SoC public and private data, should highlight the safety concerns that an applicant may meet when trying to implement a SoC on the basis of the current certification practices. This study work should be documented from concrete cases, representative of state of the art solutions.

The second objective is to propose amending recommendations for hardware certification process. The recommendations should remain in line with the philosophy of the ED80/DO254 and current hardware certification practices, and should propose a specific approach to address the safety concerns induced by the use of SoCs.

Literature Review

1. EUROCAE ED80/RTCA DO-254, “Design Assurance Guidance for Airborne Electronic Hardware,” April 19, 2000.

2. EUROCAE ED12/RTCA DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” December 1, 1992.

3. ED79/ARP 4754 Certification Consideration for Highly Integrated or complex systems. – November 1996.

4. EASA Certification Memo SW and CEH. Reference: Memo-SWCEH-002 Issue 1 Rev 2 date: 02/06/2008.

5. Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 1 Report DOT/FAA/AR-06/34 December 2006.

6. Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 2 Report June 2008.

Methodology

1. The first activity of the survey consists in selecting SoC and IP according to a specific rational.

2. Methodology for SoC public data assessment:

The purpose of the public data assessment is to provide findings about the state of SoC public data against specific design assurance objectives, identified in the guidance document ED80/DO254. The public data is:

- Data which can be freely accessed,
- Data which can be obtained through a commercial agreement (license).

ED80/DO254 provides guidance in design assurance for the development of airborne electronic hardware such that it performs its intended functions in its specified environment. This design assurance is developed from chapter 2 to 10 of this guidance document. Special considerations are given in the Appendix B for high critical safety hardware. Additional considerations that refer specifically to the use of previously developed hardware, COTS component, product service experience and tools are included in section 11.

ED80/DO254 adopts a design approach based on requirements flow from system level down to hardware elemental level according to the Design Assurance Level (DAL) defined by the system safety analysis. For a SoC component, the design, or at least a part of the design, is performed by the SoC supplier on the basis of general market requirements. Thus, the SoC is not necessarily aligned with the ED80/DO254 top-down approach.

It shall be considered that using a SoC that would have been designed according to a requirement based process similar to the one described in the ED80/DO254 should not constitute an issue in case of use in a safety critical system. Indeed, in this case, the design assurance strategy can be based on the design process defined by the chapter 2 to 10 and Appendix B.

For other SoCs, the design assurance needs to be based on alternative methods. Indeed, the information that a SoC manufacturer typically identifies as proprietary includes: detailed specifications, detailed design, source codes, specific schematics and drawings, verification results… All this data is typically required and delivered as part of a custom development program according to ED80/DO254 process.

Currently, the most popular alternative techniques being used include reverse engineering, architectural mitigation technique (ED80/DO254 Appendix B §3.1), service history (ED80/DO254 11.3, Appendix B §3.2), electronic management plan process (ED80/DO254 §11.2) and, for processing core, ED12B/DO178B activities. It could be useful to complete these alternative methods by the following key attributes identified within ED80/DO254 and pertinent to the SoC:

- They are:
    - §2.1.2 Information flow from hardware design life cycle to system development process,
    - §2.1.3 Information flow between hardware design life cycle process and software life cycle process,
    - §2.3.3 Qualitative assessment of hardware design errors and upsets,
    - §5.2.2 Conceptual design activities,
        - Item 1: High level description,

- Item 2: Major component identified, impact of unused function,

- §5.2.3 Detailed design process activities,
  - Item 2: Architectural design technique,
  - Item 4: Assessment of unused function,
- §5.4.1 Implementation objectives,
  - Item 2: The Hardware item implementation, assembly and installation is complete,
- § 6.3 Validation and verification methods,
  - 6.3.3.1: Requirement review,
    - Ensure the acceptability of the requirement,
  - 6.3.3.2: Design review,
    - Determine that the design data and implementation satisfy the requirements,
- §7.2.3 Problem reporting, tracking and corrective action,
  - All items,
- §7.2.4 Change control,
  - All items,
- §11.4 Tool assessment and qualification,
- Appendix B,
  - §2 Functional failure Path analysis,
  - § 3.3.1 Elemental analysis,
    - §3.3.1.1 Identification and definition of the elements at an appropriate level of the hardware design,
    - §3.3.1.1 Verification coverage to which each element should be verified,
  - §3.3.3 Safety specific analysis.

Thus, the strategy is to assess the public data against the above alternative methods and key attributes. Four main steps have been established to cover in a logical way all items identified above:

- Step 1: On chip cores identification and features determination,
- Step 2: Fault tolerance and fail safe features assessment,
- Step 3: Verification and design tools assessment,
- Step 4: SoC qualification assessment.

For each step, a questionnaire has been established in order to address all the selected SoCs with the same rigor.

- Step 1 - On chip cores identification and features determination.

The idea here is to try to identify the various cores or functions that constitute the SoC, and to determine their features. The processing core itself and infrastructure core are composed of several elemental hardware functions that may represent a safety risk if they are used. Furthermore it seems important to assess the degree of testability of the different cores and define if the controllability and observation of the cores can be ensured.

The second objective of this step is to determine if the available public data is relevant to allow an efficient implementation of the SoC in a safety critical application.

The following table details each topic addressed during step 1 assessment, giving: the objectives, the covered chapter(s) of the ED80/DO254, the question reference number used to address the topic (the questionnaire is given just after the table).

Step 1 topics	Objectives	ED80/DO254 key element	Question reference No
Public data identification	To identify a potential gap between the list/content of public data and the hardware life cycle data requested by the table A-1 of ED80/DO254.	§11.2.2 Item 1 Appendix A (Hardware life cycle data)	1
Features assessment – activation/ deactivation	To identify and assess the high level design concept of the SoC (functional block diagrams, architecture description). To identify how the features contribute to the hardware/system requirements, including the impact of potential unused function.	§5.2.2 (Conceptual design activities) Items 1 and 2 §5.3.2 (Detailed design activities) Item 4 Appendix B elemental analysis § 3.3.1.2 Item 3 (Unused function) ED12B/DO178B	2, 3, 4, 5, 6, 7
Internal accessibility assessment	To identify the accessibility of the internal interfaces between the different constituent of the SoC. To assess the possibility to control and observe specific internal features.	§5.2.2 Item 3 (Interface definition), Appendix B elemental analysis §3.3.1.2 Item 1	8
User guides and Errata data assessment	To assess the level of completeness and correctness of public data against the product.	§5.4.1 (Implementation objectives) Item 2 §6.3.3.1 (Requirement review) §6.3.3.2 (Design review) Appendix B elemental analysis §3.3.1.2 Item 2 (inadequacies in requirements)	9

Step 1 topics	Objectives	ED80/DO254 key element	Question reference No
Silicon errata	To identify and assess the errata related to the physical device.	§7.2.3 (Problem reporting, tracking and corrective action) Item 2, 4, 5 §5.2.2 (Conceptual design activities) Item 1.a §6.3.3.1 (Requirement review) §6.3.3.2 (Design review)	10, 11

Table 1 - Step 1assessment topics

Step 1 questionnaire:

The following questions have been used as a guideline to analyze the public data against the step 1 objectives.

1. Is there information available on the design, production, validation and verification phases from the SoC manufacturer?
2. Is it possible to identify the constituents of the processor core and to determine their features?
3. Is it possible to identify the infrastructure cores and to determine their features?
4. Are all internal identified features compatible with any kind of safety critical application?
5. Is it possible to identify the internal busses communication topology? Are their features completely documented?
6. Is it possible to isolate or disable a specific documented function or core from the rest of the SoC?
7. Is the deactivation/isolation mechanism described?
8. Is it possible to access, observe and control independently the different constituents of the processor core and the infrastructure cores?
9. Is the data of data sheets, user guides, application notes… complete and coherent with the product?
10. Are the devices errata and work arounds available?
11. Can the device errata be considered as minor regarding their safety impact?

- Step 2 - Fault tolerance and fail safe features assessment.

Failure modes and effects of SoC functions need to be analyzed to establish if the hardware architecture and implementation could comply with the system safety requirements. It should allow to determine the safety-sensitive aspects of the SoC/IPs, and their associated internal features for which design error mitigation/removal emphasis is needed.

The following table details the topics addressed during step 2 of the assessment.

Step 2 topics	Objectives	ED80/DO254 reference	Question reference No
Failure modes assessment	To identify and assess how and to which extent the failure of SoC features (random, design error and upset) may affect the SoC behavior. To identify which failure preclusion and mitigation mechanisms are implemented in the SoC?	§2.3.3 (Qualitative assessment of hardware design errors and upset) §2 Functional Failure Path analysis §5.2.2 (Conceptual design activities) Item 2 §5.2.2 (Conceptual design activities) Item 1.a §5.3.2 (Detailed design activities) Item 2 Appendix B §3.3.2 Safety specific analysis	1, 5
Memory upsets	To identify and assess which part of the SoC may be SEU/MBU sensitive, and which failure preclusion and mitigation mechanisms are implemented for upsets.	§2.3.3 (Qualitative assessment of hardware design errors and upset) §5.2.2 (Conceptual design activities) Item 1.a §5.3.2 (Detailed design activities) Item 2	2, 3, 5
Internal data transfers failure	To identify and assess potential failures on internal data transfers, and which mechanisms are implemented to preclude and mitigate such failures.	§2.3.3 (Qualitative assessment of hardware design errors and upset) §5.2.2 (Conceptual design activities) Item 1.a §5.3.2 (Detailed design activities) Item 2 Appendix B §3.3.3 Safety specific analysis	2, 5
Unused functions deactivation failure	To identify and assess the impact of a potential failure in an unused function deactivation mechanism.	§5.3.2 (Detailed design activities) Item 4 Appendix B §3.3.3 Safety specific analysis	4, 5

Step 2 topics	Objectives	ED80/DO254 reference	Question reference No
Timeout failure	To identify and assess internal features which could impact the time needed to execute a task.	§5.2.2 (Conceptual design activities) Item 1.b §2.1.3 (Information flow between hardware design life cycle processes and software life cycle processes) Item 1, 2, 3, 4	6
Microcode design errors	To identify potential failures related to microcodes embedded in the SoC	§2.1.3 (Information flow between hardware design life cycle process and software life cycle process) Item 1	5

Table 2 - Step 2 assessment topics

Step 2 questionnaire:

The following questions have been used as a guideline to analyze the public data against step 2 objectives:

1. Is it possible to identify the failure modes of the Cores and the potential effects at SoC level?
2. Are there safety mechanisms available in the SoC (i.e. internal and external bus address/data parity or ECC coverage, internal register parity, internal clock monitoring, Memory access privilege)?
3. Is there any part or function of the chip SEU/MBU sensitive and if so, is there any mechanism that allows detecting, preventing, or correcting a SEU/MBU?
4. Can we consider the deactivation of unused functions safe?
5. Can these safety mechanisms be sufficient to consider the SoC as fault tolerant or fail safe?
6. Is the SoC able to produce expected results after a guaranteed amount of time (timeout)?

- Step 3 - Verification and design tools assessment

Some SoC or SoPC require the use of specific tools (software or hardware tools) and internal chip debug functions to achieve the design and verification activities. The purpose of this step is to identify the tools and the debug functions in order to assess the risk to introduce an error in the hardware items or to not detect hardware or software errors during verification activities.

The following table gives the topics addressed during step 3 assessment.

Step 3 topics	Objectives	ED80/DO254 reference	Question reference No
Tools	To identify the need of specific tools to design the SoC. To assess the tool capability of performing the particular design and verification activities to an acceptable level of confidence.	§11.4.1 item 1 §11.4.1 item 2 §11.4.1 item 5	1, 2
On chip debug facilities	To identify internal SoC features which could contribute to hardware and/or software verification activities.	§2.1.3 (Information flow between hardware design life cycle processes and software life cycle processes) Item 2 §5.3.2 (Detailed design activities) Item 4 Appendix B elemental analysis § 3.3.1.2 Item 3 (Unused function)	1, 3, 4

Table 3 - Step 3 assessment topics

Step 3 questionnaire:

The following questions have been used as a guideline to analyze the public data against step 3 objectives:

1. Are there tools available to implement and verify the SoC (compiler, builder, debugger, SoPC design kit…)?
2. Can we do without these tools?
3. Are there debug and performance functions implemented inside the SoC?
4. Can we rely on tools and internal debug functions to not introduce an error in the design, or to not fail to detect an error?

- Step 4 - SoC qualification assessment

The ED80/DO254 contains special sections dedicated to Commercial Off The Shelf (COTS) component usage §11.2 and Product service experience §11.3.

The following table gives the topics addressed during step 4 assessment.

Step 4 topics	Objectives	ED80/DO254 reference	Question reference No
Production Track record	To identify production track record of a high quality component.	§11.2.1 Item 1	1
Quality procedure	To identify and assess quality procedures established by the SoCs manufacturers.	§11.2.1 Item 2	2
SoC qualification	To identify and assess SoC qualification procedures, which ensure component reliability.	§11.2.1 Item 4	3, 4
SoC service experience	To identify and assess SoCs service experience and pertinence.	§11.2.1 Item 3 § 11.3 Appendix B §3.2	5
Change process and problem reporting	To identify and assess the change process and the problem reporting process.	§7.2.3 §7.2.4 §7.2.5	6, 7
Obsolescence – guarantee – debug support	To identify the risk for a SoC which could become non-procurable or which could be no more supported	§7.2.5 §11.2.2 item 4	8

Table 4 - Step 4 assessment topics

Step 4 questionnaire:

The following questions have been used as a guideline to analyze the public data against step 4 objectives.

1. Can the SoC manufacturer demonstrate a track record for the production of high quality SoC devices?
2. Are quality procedures established?
3. Are there references to a SoC qualification process which establish the SoC reliability?
4. Is there qualification data available?
5. Is there any service experience record?
6. Is there a design change control process?
7. Have we the guarantee that all changes and problems are subject to customer notification?
8. Are there a guarantee of support and a guaranteed period of device production?

3. Methodology for private data assessment:

Information about the design, production, testing, and verification performed by the SoC or IP supplier may be considered as proprietary information and may require specific agreements for data exchange submitted to confidentiality. Access to such information could be useful to complete or consolidate the public data assessment.

A questionnaire has been sent to all identified SoC providers to introduce the issues related to the SoC usage within safety critical applications and to draw attention to the SoC supplier of the strategic aspect of this survey for their use in aeronautical critical applications. The objective of this questionnaire was to access mainly the willingness of SoC providers to cooperate with the aeronautical market and to know what kind of information they are ready to share. Based on their answers, the possibility to involve the SoC providers in the certification process has been assessed.

4. Methodology for recommendations:

The recommendations for certification have been built taking into consideration the conclusions from the survey made on the public and private SoC data assessment.

1. INTRODUCTION

The European Aviation Safety Agency (EASA) has mandated Aeroconseil to carry out a survey on the safety implications of the use of system-on-chip (SoC) in airborne critical applications. The goals of the survey are to identify possible safety concerns induced by the use of SoCs in safety critical applications, and to propose amendments to specific certification process to address those concerns.

This document is the result of the above stated study.

Section 2 defines terms used to address electronic components in the aeronautic industry. At this point, it is necessary to clarify the differences between COTS, ASIC, custom devices, Intellectual Property (IP)…, before dealing with the definition of a SoC. SoCs are defined and characterized at the end of section 2.

The further four sections are each dedicated to one activity of the survey.

Thus, section 3 lists and justifies the SoC providers and devices that have been selected for this survey.

The result of the assessment of the selected SoC public data is described in section 4. This section gives first conclusions regarding the use of SoCs in airborne safety critical applications.

Section 5 gives the position of the SoC providers to have a cooperative approach to give access to confidential data, and more generally, how they could be involved in the certification process.

Finally, on the basis of section 4 and 5 results, section 6 proposes amendments for certification process in order to allow the implementation of a SoC in a critical application. The recommendations integrate potential difficulties that may be met by an applicant trying to cope with these new requirements.

The survey general conclusions are presented in section 7.

2. DEFINITION - COMPLEX DIGITAL COMPONENT

2.1 COMPLEXITY CONSIDERATION

As defined by the §1.6 of the ED80/DO254, "*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 behavior.*

When an item cannot be classified as simple, it should be classified as complex."

The notion 'simple-complex electronic' is very important as the results impact the design activities which need to be done. For simple electronic hardware, it is only required to perform and document the verification and configuration management activities.

Several pre-determined criteria to classify hardware functions as simple/complex have been defined by the aerospace industry, but currently do not exist as a formal agreement.

This survey will address complex hardware functions, as no design assurance can be gained by exhaustive testing on such hardware.

The following hardware functions are considered as complex:

- Multi -Processing functions,
- PCI , PCI express, CAN, Ethernet, enhanced DMA, Rapid I/O controller, ARINC 429,
- Graphic controller,
- Ethernet transceiver.

The following hardware functions are considered as simple*:

- SERDES*,
- SPI, I2C, ARINC controller, RS232 Interface*,
- Memory controller*,
- Analog/Digital conversion*,
- Simple arithmetic function (Multiplication...).

* Care shall be taken in this classification as it could be subject to modification.

2.2 COTS FAMILY

2.2.1 Definition

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. (ED80/DO254 – Appendix C – Glossary of terms)*

The COTS is developed by a supplier which controls the design, the production, the maintenance and the commercial activities. The requirements of the customer are not taken into account by the COTS suppliers as the component is developed for multiple applications of multiple customers. The supplier selects a COTS deemed suitable for his own application.

The main benefits to implement COTS are to:

- Reduce the development costs,
- Reduce the time needed to develop an equipment,
- Use the last technology novelties.

2.2.2 Main Complex COTS

Microprocessor: An integrated circuit capable of executing a stored series of instructions from memory and writing results to memory. Evolving microprocessor architectures include concepts such as caching, pipelining, branch prediction, and other advanced features. Examples of current microprocessors used for aeronautical applications are:

- Digital signal Processor: Texas Instrument: TMS320C32
- General Purpose processor: Freescale: MPC755

Microcontroller: An integrated circuit which executes software in a specific core area and implements complex peripheral hardware elements such as for example I/O (bus controllers).

Examples of current microcontrollers used for aeronautical applications are:

- Microcontroller based on DSP: Freescale DSP56F807: Digital Signal Controller (Simple peripheral : PWM, Timer, ADC)
- Microcontroller based on DSP: Texas Instrument TMS32C6415 (complex peripheral: PCI controller…)
- Microcontroller based on General purpose Processor: Freescale MPC555 (Complex Peripheral: CAN controller

Controller/Transceiver/Bridges/Switches: Component which acts as a bridge between different bus standards or bus processors. Examples of current bridges/switches/controllers used for aeronautical applications are:

- CAN Controller: MCP2515 of Microchip
- ARINC 429 controller: HI-8583PQT10 HOLT INTEGRATED CIRCUITS
- PCI-DSP Bridge Controller: Texas instrument : PCI2040
- CAN controller: NXP semiconductor : CAN SJA 1000
- PCI controller: PLX: PLX9054
- Ethernet transceiver: Intel LXT971ALE

Graphic Processor: A Processor specifically designed to create graphical images. An example of a current graphic processor used for aeronautical applications is:

- Graphic processor: ATI M9 Graphics - AMD

2.3 CUSTOM DEVICES

- Programmable Logic Devices (PLD): A component that is purchased as an electronic component and altered to perform an application specific function. PLDs include, but are not limited to, Programmable Array Logic components (PAL), General Array Logic components (GAL), Field Programmable Gate Array components (FPGA), and Complex Programmable Logic Devices (CPLD).

- Application Specific Integrated Circuit (ASIC): Integrated Circuits which are developed to implement a function, including, but not limited to: gate arrays, standard cells, and full custom components encompassing linear, digital, and mixed mode technologies.

2.4 INTELLECTUAL PROPERTY CORE

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). Also known as virtual components, they have been classified according to three categories:

- Soft IP core: The soft IP cores are provided in a HDL form such as Verilog or VHDL. As the source code is available, they can be fully analyzed and modified by the user. For the same reason, they are subject to property concerns.
- Firm IP core: The firm IP cores are provided in a technology-independent netlist format. This allows the IP vendor to hide the critical IP details and yet allows the system integrator to perform some limited amount of optimization during placement, routing, and technology-dependent mapping of the IP block.

  Note: An IP provided in an encrypted HDL form is considered as a firm IP because the code is not readable, but the IP can be customized/optimized to a small extent.
- Hard IP core: The hard IP cores are provided in a technology-dependent physical layout format, using an industry standard language such as stream, polygon or GDSII. The hard IP cores are like black boxes and cannot be properly analyzed and/or optimized.

  Note: An IP provided in a netlist form with Place and Route constraints is considered as a hard IP. Indeed, there is no possibility to customize/optimize the IP, and its performances can be considered as constant as the place and route result is unchanged from a chip to another one.

The table hereafter summarizes the classification of the different IPs.

IP Category	Format	Properties
Soft IP	HDL (unencrypted)	Full visibility on the design Fully customizable
Firm IP	HDL (encrypted)	Customizable through dedicated tools/wizards Synthesis can be optimized to a small extend
	Netlist (without Place and Route directive)	No customization of the function Optimization to a small extent during Place and Route
Hard IP	Netlist (with Place and Route directive)	No customization nor optimization
	Physical layout format	No customization nor optimization

Table 5 - IP categories

Note that some PLD are provided with wired embedded hard IP cores. The cores can be used or not, according to the programmability capacity of the PLD. In this case, the hard IP is not provided in a physical layout format but is already hard-wired in the final chip.

2.5 SOC

A System on Chip (SoC) 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

  This category of SoC is based on standard micro-processor cores that have been completed with various hardware functions. Such components are not customizable by the user who can only configure the behavior of the device through programming registers. Unneeded functions can not be removed from the chip.

  - Examples of SoCs for general application are:
    - Texas Instrument : TMS320DM6443 Digital Media System-on-Chip (DMSoC)
    - Freescale: MPC 8270, 8541, MPC 8610….

  Note: A microcontroller could be considered as a microprocessor based SoC.

- The custom SoC (PLD/ASIC)

  They are based on standards PLD, or designed as an ASIC.

  As any PLD or ASIC, they are fully defined by the designer who chooses all the necessary functions to integrate in the chip, and mainly the type of micro-processor core (PowerPC, ARM, 8051, SPARC, proprietary…). The various functions are generally available as IPs, but they can also be own made.

  Potentially, any PLD can implement a SoC. Practically, only the biggest devices are used due to the amount of logic required to implement an efficient micro-processor core. SoCs based on a PLD are also called System on Programmable Chip (SoPC).

  - Examples of PLD devices generally used as SoPCs are:
    - Xilinx: Virtex and Spartan platform
    - Altera: Stratix and Cyclone platform
    - Actel: Igloo and ProASIC platform

  Some SoPCs are proposed with one or several hard IP cores already wired on the chip. Those IPs may be µProcessor cores and/or any hardware functions like standard interfaces, calculation blocks…

  Thus, according to the type of the PLD embedded resources, it is possible to build a SoPC with more or less glue logic, more or less firm/soft IP and more or less self made development.

  The figure hereafter illustrates the four different possibilities to build a SoPC from a PLD.

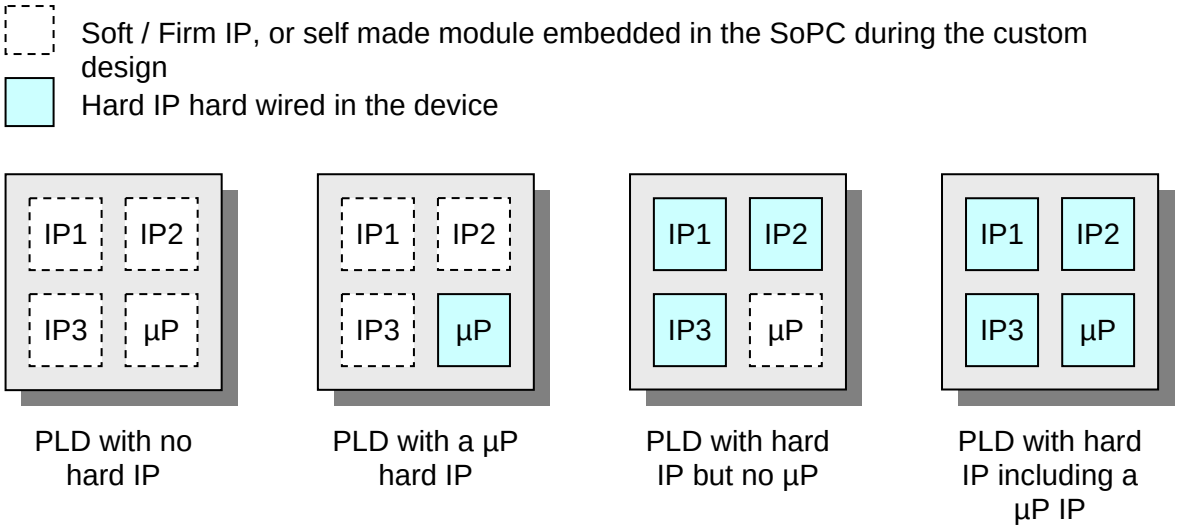


Figure 1 – From the PLD to the SoPC

The Figure 1 highlights the fact that the assessment of a SoPC against design assurance standards comes to the same as assessing the IP that composes the SoPC, and the way they are embedded in the global design.

The figure hereafter summarizes the COTS and custom devices repartition. The grayed box identifies the perimeter of the survey.

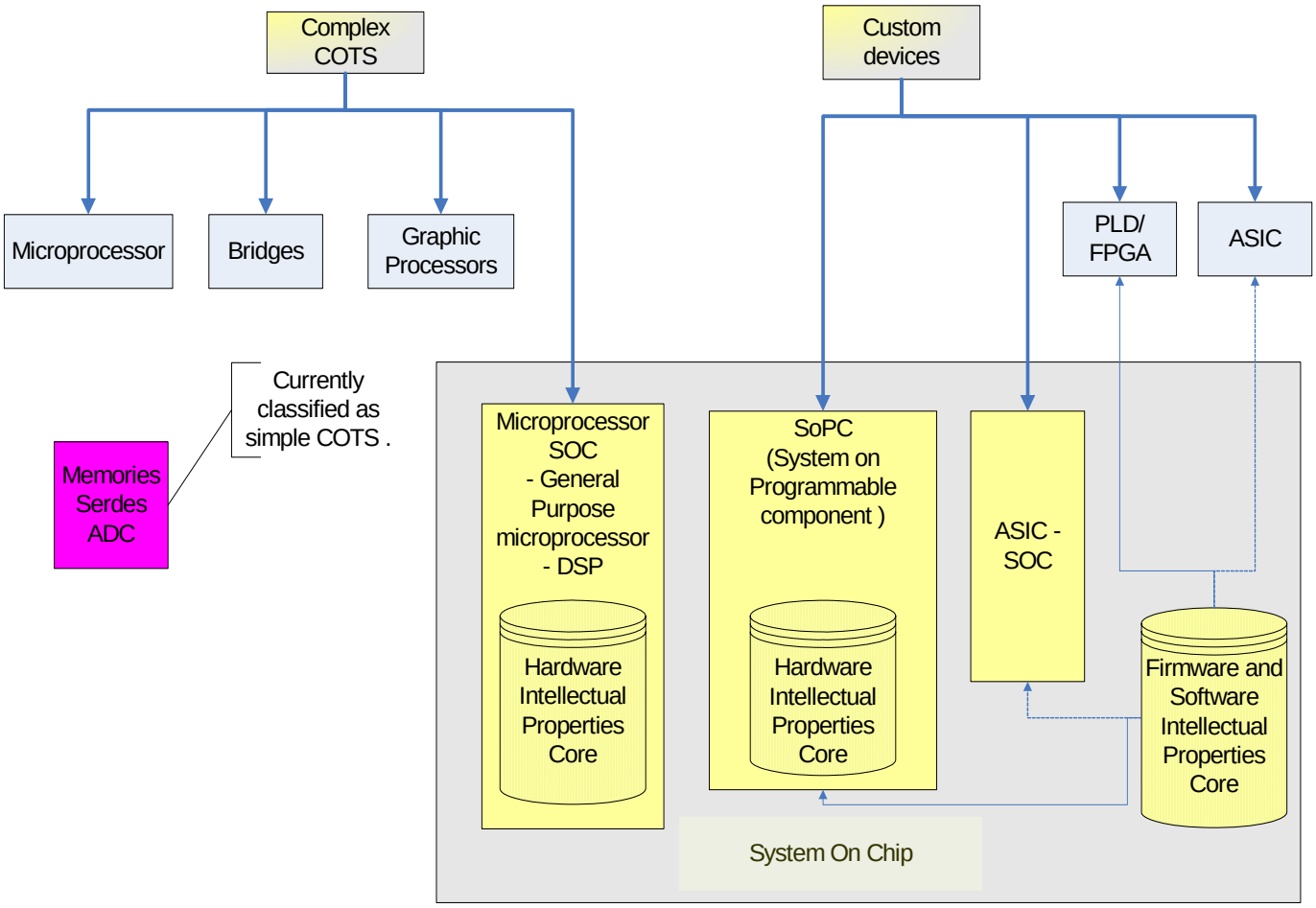


Figure 2 – COTS and Custom devices repartition

It must be noted that an ASIC design flow is quite similar to a PLD design flow; it differs mainly at the back end phase. Thus, the assessment of an ASIC SoC design against design assurance standards will be carried out through the assessment of the SoPC.

3. SOC SELECTION FOR THE SURVEY

3.1 MICROPROCESSOR – SOC

3.1.1 Rational for selection

Industrial Rational

The Microprocessor SoC survey will focus on Freescale and Texas Instrument products. Microprocessors of these providers have been largely used on last Aircraft programs and are now integrated inside the SoC. The transfer from Simple Processor architecture to SoC architecture is cost efficient as the operating system running on both chips is compatible.

Example - computing processing board architecture evolution: e600 core is implemented in the 2 components which will make easier the OS implementation and certification tasks, according to ED12B/DO178B.

CPU Board Architecture- Based on Freescale processor MPC 74XX (e600 core) The Bridge in Yellow is a customized design	Trend: New CPU Board Architecture Based on Freescale SOC MPC 86XX (e600 core) The Bridge function is integrated in the MPC 86XX

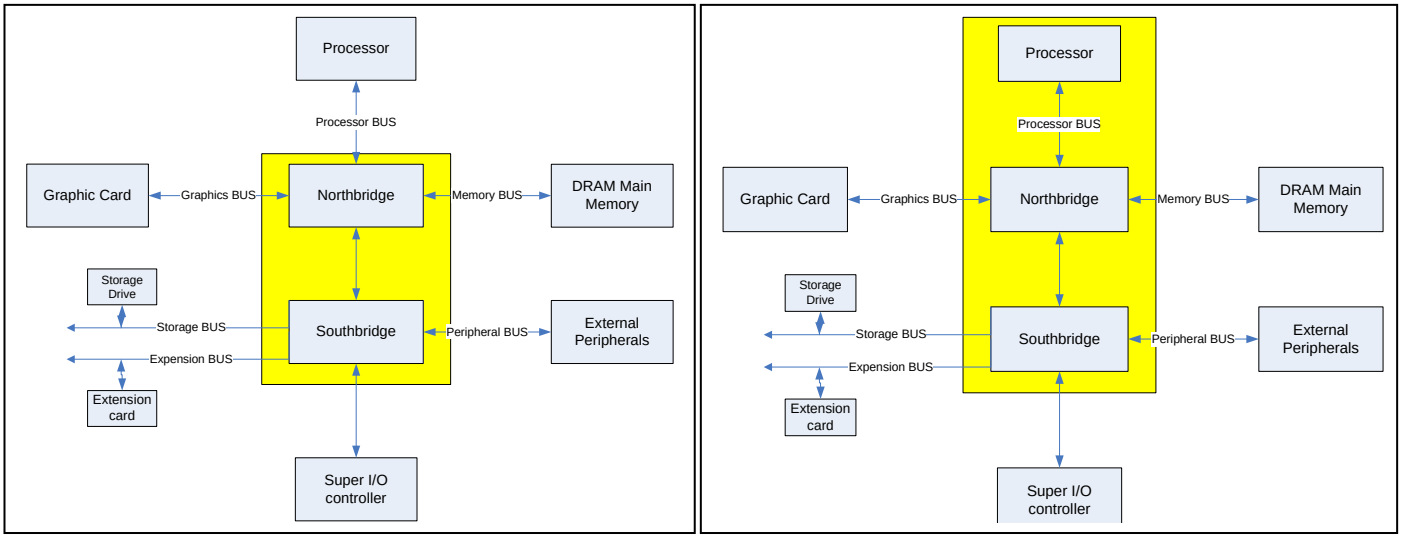


Figure 3 – CPU boards architecture evolution

Technical rational

Hereafter are listed the technological trends which will or are starting to be used for aeronautical applications.

- Smarter memory controllers

Memory controllers and bridges are also combined with multiple cores on single pieces of silicon with dramatic effects. Memory subsystems have historically been a bottleneck in high performance processing systems. Recent developments in memory technology, including the introduction of Double Data Rate 2 (DDR2) interfaces, have significantly improved performance. DDR2 provides transfer rates up to 667 MHz compared to the 133 MHz of Single Data Rate (SDR) technology.

- High bandwidth on-chip interfaces

Getting data in and out of the processor is another focus area for system performance improvement. Now systems requiring high bandwidth data transfer need to use on-chip high-bandwidth pipes enabling direct connection of Gigabit Ethernet, Serial RapidIO and PCI Express from the backplane to the processor with no external devices.

- Direct Memory Access (DMA) and Enhanced Direct Memory Access (EDMA)

In many applications, the primary bottleneck in system performance is the efficient movement of data. Various innovations have been introduced to processor architectures that offload many I/O or data movement tasks from the processor core, allowing it to concentrate on signal processing tasks. The direct memory access (DMA) engine is a critical component of most high-performance processors. Instead of having to explicitly access memory or peripherals, the processor can configure the DMA engine to access the on- and off-chip resources, and facilitate the transfers between them.

- Dual core processors

System designers are moving towards multi-core processor architectures rather than higher frequency devices to enable higher system performance while minimizing increases in power consumption. Dual core microprocessors, originally conceived for computationally intensive applications such as servers, are now being designed and could be deployed across a range of embedded applications.

- Parallel processing with Single Instruction, Multiple Data (SIMD) engines

SIMD engines enable highly parallel operations, allowing for simultaneous execution of multiple operations in a single clock cycle by means of instruction level execution units that operate concurrently with existing integer and floating-point units.

Based on industrial and technical rational, the following SoC microprocessors have been selected:

- Freescale MPC 8641 D, based on dual general purpose microprocessor
- Texas Instrument TMS32C6415, based on Digital Signal Processor

3.1.2 Freescale MPC 8641 D features

Freescale processors, built on Power Architecture technology, have steadily evolved in performance, integration and capability since the first product launched in the mid –1990's.

Today the MPC8641D is one of the most powerful SoC provided by Freescale implementing 2 high-performance e600 microprocessor cores and a complete set of peripherals to answer the needs of networking and industrial market.

MPC 8641 functional overview:

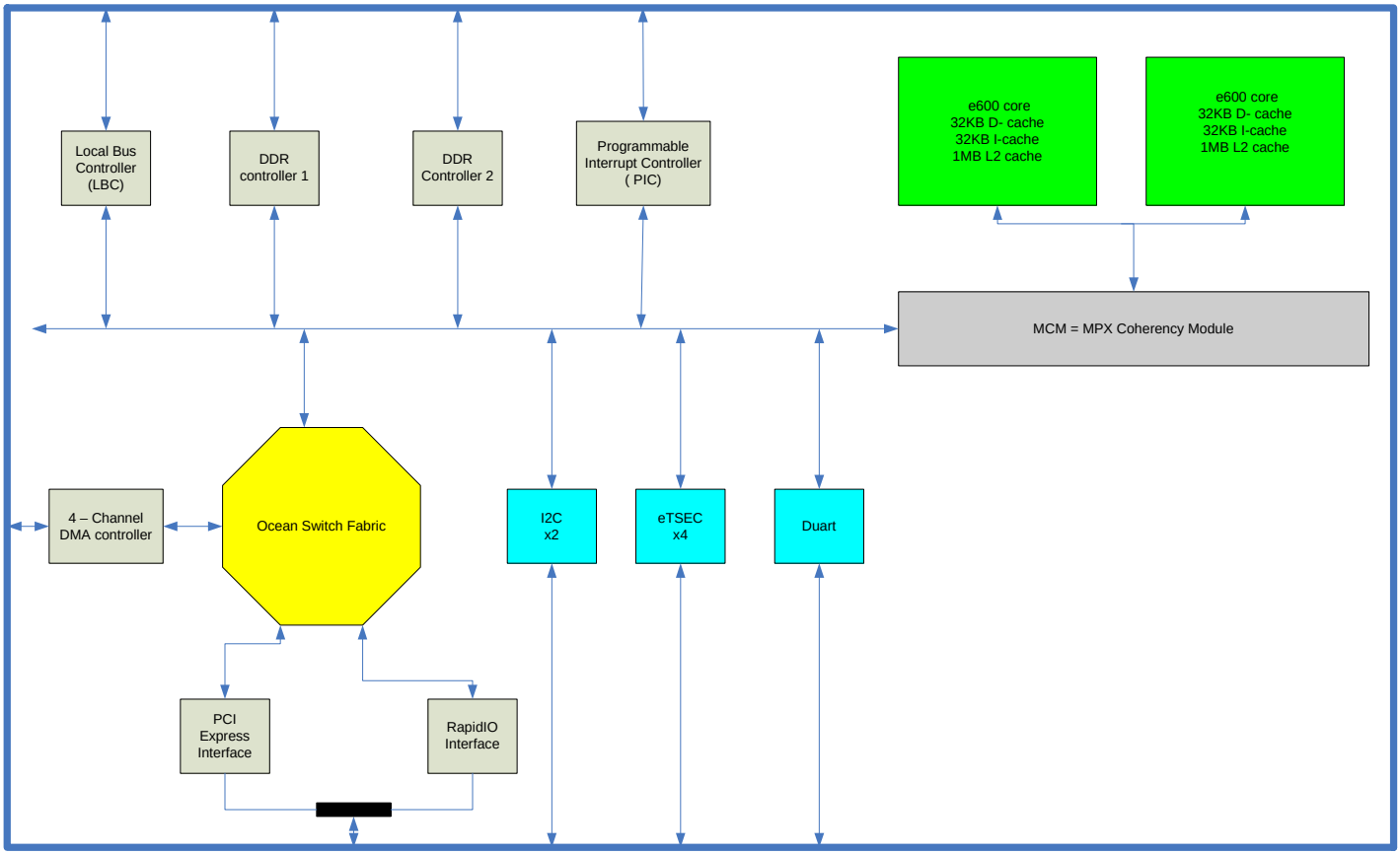


Figure 4 – MPC8641D functional overview

NB: The Figure 4 is a functional overview and is not necessarily representative of the real implementation of the MPC 8641D.

Using the reference documentation available from the manufacturer, the following on-chip cores have been identified inside the 8641 D:

Processor Core

- 2 e600 processor core (Pipeline 7 stages, including SIMD, branch prediction…)
- On-chip cache (L1, L2)

- e600 Coherency Module (MCM – MPX Coherency module)

Infrastructure Core

- OCeaN on-chip network,
- DMA controller: 4 General-purpose channels. Transfers can be requested by an external device,
- RapidIO interface controller: Is able to initiate transaction on the platform Bus through the OceaN switch,
- Peripheral component interconnect-extended (PCI-X) controller: Is able to initiate transaction on the platform Bus through the OceaN switch,
- Embedded Programmable Interrupt Controller,
- Integrated circuit (I2C): Do not implement DMA channels. The e 600 is in charge of emptying the receive FIFO and feeding the transmit FIFO,
- Double data rate (DDR) memory controller,
- General-Purpose Chip-Select Machine,
- Ethernet™ controllers: Each eTSEC has 2 dedicated DMA channels, one for reception and one for transmission,
- Dual Universal Asynchronous Receiver Transmitter (DUART): Do not implement DMA channels. The e 600 is in charge of emptying the receive FIFO and feeding the transmit FIFO,
- Local Bus Controller (LBC),
- JTAG boundary scan.

Clock

- e600 clock: 1GHz,
- Syst clock: 66Mhz,
- Bus Platform: 333 Mhz.

Miscellaneous

- Device performance monitor.

3.1.3 Texas Instrument: TMS320C6415T feature

The TMS320C64157 is the highest performance fixed-point DSP generation in the TMS320C6000™ DSP platform. The component is intended to answer to the telecommunication market needs.

TMS 320 C6415T overview

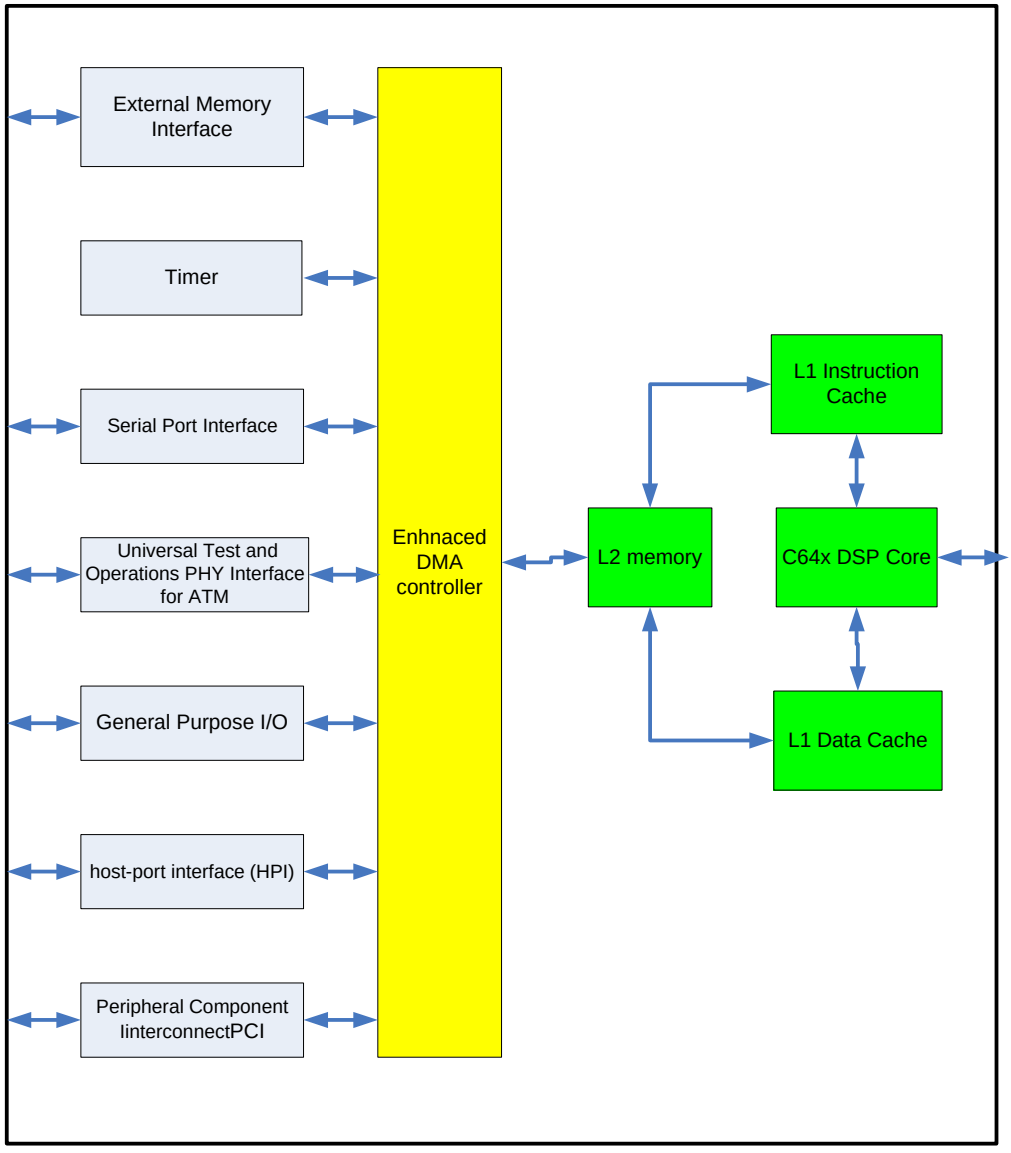


Figure 5 – TMS320C6415T overview

Using the reference documentation available from the manufacturer, the following on-chip cores have been identified inside the TMS320C6415T.

Processor Core:

- Highest-performance fixed-point DSP generation in the TMS320C6000 DSP platform running at up to 1 GHz. The C64x is code-compatible,

- Based on the second-generation high-performance, advanced very-long-instruction-word (VLIW) architecture (VelociTI.2) developed by Texas Instruments,
- 2 levels of cache.

Infrastructure Core:

- Three multi-channel buffered serial ports (McBSPs),
- A 8-bit Universal Test and Operations PHY Interface for Asynchronous Transfer Mode (ATM) Slave [UTOPIA Slave] port,
- Three 32-bit general-purpose timers,
- A user-configurable 16-bit or 32-bit host-port interface (HPI16/HPI32),
- A peripheral component interconnect (PCI),
- A general-purpose input/output port (GPIO) with 16 GPIO pins,
- Two external memory interfaces (64-bit EMIFA and 16-bit EMIFB), both of which are capable of interfacing to synchronous and asynchronous memories and peripherals,
- Enhanced Direct-Memory-Access (EDMA) Controller (64 Independent Channels).

3.2 SOPC / IP

3.2.1 Rational for selection

Industrial Rational

This part of the survey dedicated to the SoPC and IP will mainly focus on Xilinx, Altera and Actel PLDs. Indeed, the first two are world leaders in PLD, and their devices are generally used in aeronautical applications requiring a high computing power. The third one is the leader in PLD for aeronautical applications and especially for safety critical applications.

Technical rational

A complex PLD, fully developed by the applicant, can be designed according to the ED80/DO254 and thus, should not be subject to the ED80/DO254 compliance issues. The certification problems may arise when using IPs where there is no visibility at all on their design methodology, quality of produced data, configuration management…

That is why this part of the study, dedicated to the SoPC, will mainly focus on the different IPs which allow to implement a SoC in a PLD.

Moreover, the assessment of the selected IP against design assurance standards will necessarily take into account the various integrated tools that ease the use of IPs nowadays. Indeed, SoPC providers propose integrated tools that allow implementing and customizing their SoCs very easily. It is even possible to configure and implement a complex SoC without writing a single VHDL code line. These “pushbutton” solutions contribute to the lack of visibility of the design and may constitute a certification issue.

The survey will mainly focus on the micro-processor core IPs as they are the heart of the SoPCs. Nevertheless, attention will be paid to other most common IPs which come with the selected micro-processor core, because they are part of the SoPCs and may contribute to certification issues.

3.2.2 Hardware micro-processor IP

In this category, 2 kinds of sub categories can be identified:

- The silicon hardware IP embedded in the silicon of the PLD.

At this time, only Xilinx proposes a SoPC based on a silicon hardware micro-processor IP. With the Excalibur platform, Altera had introduced such a chip a few years ago, but those devices, although still sold, are not recommended for a new design. Thus, in this category, the only addressed devices will be:

  - The Virtex 5 FXT platform which embeds a PowerPC 440 hardwired core,
  - The Virtex 4 FX platform which embeds a PowerPC 405 hardwired core,
  - The Virtex II Pro platform which embeds also a PowerPC 405 hardwired core.

- The Hardware IP delivered as a netlist already placed and routed.

The Cortex M1 (Actel) IP is an ARM micro-processor core. Even if ARM processors are not yet widely used in aeronautical applications, ARM is a leader in embedded processors. Thus it seems interesting to address this IP as part of the survey.

3.2.3 Firmware micro-processor IP

In this category, 2 kinds of sub categories can be identified:

- The IP dedicated to a specific platform device or a specific PLD brand.

Those IPs are generally supplied by the PLD provider (or a partner) and are optimized in size and efficiency for a specific platform. For the study, the MicroBlaze from Xilinx has been selected. This IP has been selected because it is widely used on Xilinx platforms.

- The generic IP which are device independent.

Those IP are generally multi-platform and are not necessary optimized for a specific target.

Large number of such IP providers, proposing micro-processor cores, can be found on the market. The 8051 core from Digital Core Design has been selected because this company has signed a partnership with Xilinx and Altera. Moreover, the 8051 architecture has been used for several years in aeronautical applications and then, such an IP may be widely used in the future as a replacement solution of obsolete circuits.

3.2.4 Software micro-processor IP

As mentioned earlier, such an IP should not represent an issue regarding the certification. Indeed, as the source is available, the applicant can implement a full ED80/DO254 process.

However, it has been decided to choose an “open source” software IP in the frame of this survey. Indeed, it will be interesting to identify what kind of data can be provided with such an IP, and to assess the possibility to reduce the workload to reach the certification, taking benefits of “open source” documentation. The selected open source software IP is the Leon3-FT. It is the “Fault Tolerant” (SEU tolerant) version of the Leon3 core, which is widely used in spatial applications.

Moreover, the NIOSII-SC from HCELL Engineering will be also addressed. This IP has been selected because it is the “Safety Critical” version of the well known NIOSII core from Altera. It is announced as compliant to the ED80/DO254.

3.2.5 Development tools

All three main PLD manufacturers propose a set of tools dedicated to the development of their devices. These tools include a library of IP (hard, firm or soft, free or licensed) from which it is possible to build a SoC. The designer has just to select the IP he/she wants to embed into the SoC, parameterize them (size, speed, options…), select the communication means between the IP (bus type, bus controller…) and let the tool generate all the files (netlists, constraint files…) needed to implement the SoC in the target device.

Thus, the assessment of the µProcessor core of each SoPC providers will come with the assessment of the following tools:

- The set of tools ISE / XPS for Xilinx devices,
- The SoPC builder tool for Altera devices,
- The set of tools Libero / CoreConsole for Actel devices.

3.3 SELECTION SUMMARY

The table hereafter summarizes the SoCs and SoPCs selected for the survey.

SoC/IP Type	SoC provider	Name	Type
Microprocessor – SoC	Freescale	MPC 8641D	COTS device
	Texas Instrument	TMS320C6415T	COTS device
SoPC	Xilinx	Virtex 5 FXT, Virtex 4 FX, Virtex II Pro	Hardware IP
	Actel	Cortex M1	Hardware IP
	Xilinx	MicroBlaze	Firmware IP
	Digital Core Design	8051	Firmware IP
	Altera	NIOSII-SC	Software IP
	Gaisler	Leon3-FT	Software IP
SoPC Tools	Xilinx	ISE / XPS	Development Tools
	Altera	SoPC builder	Development Tools
	Actel	Libero / CoreConsole	Development Tools

Table 6 - Selection summary

4. PUBLIC DATA ASSESSMENT RESULT

The strategy used to perform the public data assessment has been presented in section "Methodology" page 16.

4.1 SOPC / IP

The following sections summaries the findings on microprocessor IPs.

Note: Some SoPC can be implemented in PLDs which configuration is subject to SEU/MBU. The SEU/MBU sensitivity of the PLD configuration should be addressed by the designer of any PLD and is out of the scope of this survey.

4.1.1 Step 1 - On chip cores identification and features determination

The results given in this section rely on the analysis of the public data available for the selected IPs. They are mainly:

- Data sheets,
- User manuals,
- User guides,
- Application notes,
- Quality manuals,
- Design data (for the NIOS II SC).

4.1.1.1 Features assessment – activation/deactivation

It must be noticed that the complexity varies a lot from IP to IP. The selected processor cores go from a simple 8 bits controller, with no complex feature, to a 32 bits controller with cache, MMU, dynamic branch prediction.

Generally, the selected processor IP documentation allows to identify the internal architecture of the cores and their main constituents. Thus it is possible to have global comprehension of the IP core behavior. In the same way, the internal communication mechanisms are well described in most cases.

Very often, it is possible to deactivate or isolate specific functions. This is especially true for firmware and software IPs which can be parameterized through generic parameters or input port programming. Nevertheless, the mechanisms behind those parameterization means are not documented and it is difficult to determine if a parameter acts as a reset, a clock disable, a logical disable or induce a modification of the source code and/or the netlist of the IP.

The following main features can be found in the selected IPs. All the features are not necessarily implemented in all the IPs.

- Pipeline and branch prediction unit (dynamic or static)

The pipeline mechanism sequences the fetch, the decoding and the execution of each instruction in several stages (3 to 7 according to the device). Each stage treats or executes a part of the instruction.

On a branch instruction, the core must wait for the end of the execution of the current instruction to know what will be the next instruction. In order to avoid a break in the pipeline sequence, the branch prediction module determines speculatively what will be the next instruction. At the end of the branch prediction, it determines if it was right or not. In the first case, time has been saved, in the other case, time is lost because the core shall discard the content of the pipeline and fill it with the right instructions before being able to continue running the program.

Depending on the IP, the branch prediction mechanism can be static or dynamic. A static mechanism means that the speculation is encoded in the branch instruction op-code. On the other hand, while running the program, the dynamic branch prediction unit statically measures for each branch instruction the percentage of branch taken against branch not taken. Those statistics are used to maximize the chances to perform the correct speculation.

Generally, the pipeline and the branch prediction unit can not be disabled.

- Instruction cache and Data cache

The cache is an internal fast memory that mirrors a part of an external slower memory. A cache for the instructions (Instruction cache) and a cache for the data (Data cache) are usually found. The cache unit implements algorithms that dynamically manage the cache content and ensure the consistency between the cache and the external memory it is supposed to mirror.

Generally, the cache itself can not be disabled, but the user can choose to not use it for a part or the entire address space through a software register. Nevertheless, the mechanism behind this software register is never documented.

- Memory Management Unit (MMU)

The MMU allows to divide the whole addressing space into several smaller addressing spaces. It is also able to translate virtual addresses into physical addresses (i.e. to map the software addresses to physical addresses that cope with the physical implementation of the memories), and to protect each addressing space to avoid a corruption by an unauthorized thread.

Generally, the MMU functions can be disabled through a software register, or not implemented for customizable IPs. Nevertheless, the action of these registers or parameters is not documented and the disabling mechanisms are not described.

- Processing unit extension

Some IPs allow the designer to extend the native microprocessor instruction set with custom instructions. The designer can define new op-codes that are decoded by the microprocessor, but executed by an additional core (homemade or standard IP core), that can be seen as a co-processor, connected to a dedicated interface.

The processing unit extension is often used to add to the microprocessor hardware processing capabilities, such as Floating Point Unit (FPU), that would penalize the performances if emulated in software.

Generally, the processing unit extension features are optional functions and can be disabled through software registers or parameters. Nevertheless, the action of these mechanisms is not documented.

- Bus controllers

All the selected IPs implement at least one bus controller which manage and arbitrate the communication means of the microprocessor core. A wide range of bus characteristics can be found from IP to IP, and thus, the complexity of the bus controller varies also a lot. Those buses can be:

  - Internal (for internal communication between different functions of the IP) or external (to allow the communication outside the IP),
  - Based on a proprietary technology, or to the contrary based on a standardized protocol,
  - Multi-master and/or multi slave, or point to point,
  - Dedicated to memory access or dedicated to peripheral access.

When unused, a bus controller can be generally disabled or not implemented through registers or parameters but the mechanisms behind that are not documented. On the other hand, some bus controllers can not be disabled and their interfaces can be left unconnected if unused. Here again, the internal mechanisms are not described and it is difficult to determine the potential impact of such an unconnected interface on the rest of the processor core.

- Debug module

The debug modules allow the designer to interact with the processing core in order to perform software and/or hardware debug. The functionalities of the debug modules depend on the device, but the following common functions can be found:

  - Setting breakpoints in the normal program to generate specific interruptions,
  - Using a JTAG port to stop and control the microprocessor core. It is then possible to stuff an instruction in the normal program, to modify any internal register or memory, or to run the program instruction by instruction (stepping mode),
  - Tracing out the instruction stream executed by the processor to monitor in real time which the current instruction.

Here again, the functions of the debug modules are describes but the mechanisms are not.

- Interrupt Controller

The Interrupt Controllers manage the interruption sources and force the core to jump to a specific address to serve the interruptions. They allow to prioritize, to enable or to disable the different interruption sources.

Generally, the controllers themselves can not be deactivated, but the various possible interruption sources can be enabled/disabled through software registers.

4.1.1.2 Internal accessibility assessment

The internal accessibility to a specific function depends on the IPs architecture and on the nature of the function.

Generally, the internal functions propose at least one external interface that allows to monitor its behavior. Thus, it is possible to correlate the software behavior with the external interface behavior to stimulate and monitor the simple functions. It is the case for the interrupt controllers, the simple bus controllers or the processing unit extensions. In this context, the debug modules functionalities can be useful.

Nevertheless, some functions too deeply embedded in the processor core remain not accessible and can only be indirectly observed. It is the case for the cache, the pipeline and the internal communication busses.

4.1.1.3 User guides and Errata data assessment

Generally, the IPs come with data sheets, user guides and/or user manuals that allow to implement the IP at hardware and software level.

The following information can be found with more or less details:

- IP features description,
- Pinning of the IP,
- Parameters that allow to customize the IP,
- Software registers,
- The instruction set architecture.

The IP documentation is sometimes completed by application notes and knowledge databases. Moreover, some IPs are the object of errata sheets that refer to errors in the documentation, or core unexpected behavior or un-reached performances. All this data can be used by the designer to ensure a correct and safe implementation of the IP.

Nevertheless, the amount of information and the level of detail vary a lot from IP to IP. Some are described very precisely and can be implemented correctly, while others are more lightly addressed and require the designer to perform additional testing activities to compensate the documentation ambiguities.

Moreover, no formal process that would ensure the consistency and the exhaustiveness of the published data against the actual chip has been identified. Thus, even in the case of a complete and precise set of documentation, a designer should consider potential errors in the documentation.

In the same way, a lack of published errata sheets should not be assumed as a lack of errors in the documentation or in the IP. Indeed, it is some time easier for a SoPC provider to release a new version of the IP instead of publishing errata sheets.

4.1.1.4 Silicon errata

This section concerns only hardware IPs already hardwired in the PLD (PowerPC IPs embedded in Xilinx devices).

The analysis of the published silicon errata has shown significant bugs of the processor IPs that could have a safety impact if the work around is not implemented.

4.1.1.5 Step 1 conclusion

Generally, the constituents, the main features and the main mechanisms of the IPs are identified and explained in their documentation. In most of cases, the documentation may be sufficient to implement the IP in a non safety critical application. Nevertheless, some concerns may jeopardize the good behavior of the IP:

- The action of the parameters that allow to customize the IP is not documented,
- The action of the software registers and the input ports programming that allow to disable specific functions is not documented,
- Some constituents of the IP can not be directly accessed and thus can not be fully assessed,
- The potential errors and potential lack in the documentation may lead to an inappropriate implementation of the IP.

4.1.2 Step 2 - Fault tolerance and fail safe features assessment

The processor cores functionalities are described in the literature, but the way in which they are actually implemented is poorly explained or even not at all. Then, potential failure modes can be just imagined on the basis of the explained expected behavior. Thus, in the following sections, the potential failure modes are not explicitly stated in the IP documentation but are extrapolated from the described behavior.

4.1.2.1 Failure modes assessment

The following failure modes list is not exhaustive but highlights potential failure modes of the main features of the selected IPs.

- Pipeline / Branch prediction

Considering the complexity of the pipeline and branch prediction functions, it should be assumed that a design error of these functions may lead to an undetected misbehavior. Indeed, they are used to speed up the instructions fetch, decode and execute processes and implement complex algorithms that:

  - Modify the execution order of the instructions,
  - Anticipate the result of a condition instruction to start fetching and decoding the following instructions.

The potential failure modes of this function are therefore:

  - Incorrect order of instruction execution,
  - Undetected bad prediction which leads to take an incorrect program branch,
  - Access to an incorrect memory location due to the speculative mechanism.

Those failure modes are not documented and no preclusion or mitigation internal mechanism has been identified.

- Cache

The cache is a fast access internal memory which mirrors a part of an external slower memory. Its potential failure modes are:

  - Incorrect refresh of the cache content or of the external memory, which leads to an inconsistency between the cache and the external memory,
  - Corruption of internal data due to SEU/MBU.

No preclusion or mitigation mechanism has been identified for the first issue. Nevertheless, it is possible to limit or inhibit the use of the cache.

The upset issues are addressed in some devices through specific mechanisms (cf. §4.1.2.2).

- MMU

The MMU allows to define several memory spaces and to translate the virtual memory spaces into a physical memory space. The MMU contains a TLB (Translation Lookaside Buffer) that is used to describe the various memory spaces. These TLBs are generally implemented in SEE sensitive memories.

The MMU potential failure modes are:

- Access to an incorrect memory location due to an error in translation mechanism,
- Access to an incorrect memory location due to a corruption of TLB data by SEU/MBU.

No preclusion or mitigation mechanism has been identified for the first issue. Nevertheless, it is generally possible to disable the MMU function through a software register.

The upset issues are addressed in some devices through specific mechanisms (cf. §4.1.2.2).

- PUE (Processing Unit Extension)

This module is tightly coupled with the heart of the micro-processor architecture. Each fetched instruction is fed to both the decode module of the processor and the decode module of the PUE. According to the instruction, the PUE may request to the processor an access to an internal resource (global purpose registers for example), and may return a result to the processor (a computation result or a status). If a PUE instruction returns a result to the processor, the PUE stalls the normal instruction pipeline until the instruction has been executed by the user logic attached to the PUE.

Even if not explicitly described, it must be assumed that the processor core implements mechanisms able to manage the parallelism of both processing flows.

The identified potential failure modes due to a design error are therefore:

- The instruction is badly decoded by the PUE. It is interpreted like having to be treated by the PUE while it should not. On the opposite, the instruction which could not be taken into account by the PUE should be treated.
- The PUE accesses the incorrect internal resource (in read or write),
- The PUE indefinitely stalls the normal instruction pipeline.

The above failure modes are not addressed in the documentation and neither a fault tolerant feature nor a fail safe mechanism has been identified.

No possible mean of mitigation has been identified. Indeed, the interfaces between the PUE and the processor are hidden in the IP and are not accessible for the user. Then, it is not possible to add a mitigation module that would spy and control the behavior of the PUE. Moreover, an exhaustive test of the PUE seems to be difficult or even impossible to perform. Indeed, such a test should take into account:

- All the standard instruction sets completed with the instruction assigned to the PUE,
- All the combination of instruction sequences,
- The fact that an interruption may occur at any time,
- The different possible states of the user logic attached to the PUE.

If unused, the PUE function can be generally disabled through a software register or not implemented through a parameter. Nevertheless, the action of this register/parameter is not documented (simple logical inhibit, software reset, physical disconnect…). As a result it can not be ensured that disabling the PUE module is enough to compensate a potential design error and avoid misbehavior of the processor core.

- Bus controllers

Bus controllers are more or less complex according to the functionalities of the bus (multi-master, multi-slave, burst access…). Nevertheless, the potential failure modes of communication busses are all the same:

- o Modification of the transferred data during the transaction,
- o Incorrect memory location access,
- o Never ending transaction, or a too long transaction due to an arbitration problem or handshaking issue.

The microprocessor cores provide no means to preclude or mitigate above potential failures. Nevertheless, most of the identified communication busses offer a communication path with the outside of the microprocessor core. The designer can then implement, in the PLD logic, user defined mechanisms that allow to monitor transactions between the microprocessor and its peripherals and to manage a potential failure. In the same way, the user can implement software mechanisms that periodically perform predefined transactions, whose expected results are known to monitor the good behavior of the busses.

- Interrupt Controller

The interrupt controller manages and arbitrates the various interruption events and makes the program jump to a vector address to execute the specific code dedicated to the treatment of the event.

The potential failure modes are:

- o A jump to an incorrect vector that causes a bad treatment,
- o An incorrect save and/or restore of the microprocessor context before the interruption,
- o An incorrect resume of the instruction that has been stopped to serve the interruption while it was not completely executed.

No preclusion or mitigation mechanism has been identified for the above potential failures.

- Debug Logic

As explained in section 4.1.1.1, some debug functions are very intrusive in the microprocessor core and may lead to corrupt the instruction stream. As debug functions should be used for set up purpose only, the main potential failure modes in operation are:

- o Unexpected activation of the debug module,
- o Unexpected modification of the instruction stream or of any value in the microprocessor environment.

The debug modules can be disabled or not implemented by software registers or customization parameters. Moreover the JTAG interface, used to control the microprocessor core from external pins, can be hard disabled, by not providing its main clock for instance. Nevertheless, the mechanisms behind those 2 means of deactivation are not documented, and it can not be ensured that the debug module never disturbs the normal behavior of the microprocessor core.

Note: Verification concerns when using the debug module are addressed in section 4.1.3.

4.1.2.2 Memory upsets

Even if not implemented in a SEU sensitive SRAM based PLD, a microprocessor IP can embed functions intended to be mapped on an internal PLD memory that may be SEU sensitive.

Some IPs implement mechanisms such as parity or triple vote in order to manage upsets on sensitive functions.

Nevertheless, some functions are not protected against SEU/MBU and may lead to an unexpected behavior.

- Dynamic branch prediction unit uses a BHT (Branch History Table) to draw up statistics on taken and not taken branches. The statistics are used to dynamically anticipate a specific branch. Since it is a memory, this BHT is most probably SEU/MBU sensitive (it is not documented). In case of corruption of the BHT, the branch prediction unit may just fail when anticipating which branch to take, which would then impact on the performance of the system. But a BHT corruption may also lead to a misbehavior of the branch prediction unit and then to a corruption of the execution stream.
- Cache and TLB embedded in the MMUs are also generally implemented in SEU sensitive memories. An upset on such a memory may lead to an unexpected behavior.
- Bus controllers often embed FIFO memories used to stall data when the bus is busy or to move the data from one clock domain to another one. Depending on the technology used to implement those FIFO, they may or may not be subject to SEU/MBU disruptions. Unfortunately, no information on potential embedded FIFO and on their technology has been found, and thus on potential SEU impacts on internal communications.

4.1.2.3 Internal data transfers failure

Most of the selected IPs do not implement a complex internal bus topology. Indeed, they implement only communication buses with at least one external interface that allow to monitor its behavior (cf. §4.1.2.1).

A single IP embeds a complex internal bus topology that could lead to undetected internal data transfer failures in case of design errors. Indeed, it involves several busses of different types that share a common communication node. The potential failure modes of this feature are the same than the failure modes of the bus controllers listed above. Nevertheless, some of theses buses offer no external interface, and thus, can not be monitored or mitigated.

4.1.2.4 Unused functions deactivation failure

According to the nature of the IP, unused functions can be deactivated, through a software register of an input port programming, or even not implemented through a generic parameter in the IP entity.

The registers and the input ports are well documented but generally their action is poorly or not at all described. It is not described if they act as a reset, as a clock inhibitor or as a simple logic disable. As a consequence, it should be assumed that a design error on the deactivation mechanisms may lead to an inadvertent activation and then may constitute a safety risk.

In the same way, the mechanisms related to the generic parameters are not explained. If the unused functions are actually removed from the IP, the unused functions should not constitute a safety risk. On the other hand, if the unused functions remain in the IP and are just logically inhibited, it should be addressed as a potential safety concern.

4.1.2.5 Timeout failure

Potential timeout failures should be considered for IPs that embed features such as cache, MMU and branch prediction that may impact the execution time of the software. Indeed, the lack of precise information on these mechanisms avoids an exact execution time computation, and leads the designer to inhibit such mechanisms or to overestimate the execution time but with no assurance that margins are sufficient.

It is also the case for IPs whose architecture does not ensure a constant execution time. For example, the debug module of one of the selected IP interacts on the main communication bus, and on particular conditions, may modify the time needed to access an external memory.

4.1.2.6 Microcode design errors

No microcode has been identified in any of the selected IP. On the other hand, the absence of microcode is not formally stated in the public data and the IPs may include microcode without customer notification.

4.1.2.7 Step 2 conclusion

In most cases, potential failure modes are not documented and can only be supposed from the public data analysis. Almost all the constituents may have a critical impact on the SoPC behavior in case of design errors. At the same time, the lack of internal visibility and connectivity make it difficult, not to say impossible, to perform characterization activities which would give confidence in the IP reliability. In this context, the cooperation of the IP provider seems to be essential to get IP design data or to get some support to implement efficient mitigation or preclusion solutions.

The selected IPs are not equal regarding SEU sensitivity. Some of them embed no SEU sensitive functions, some others embed SEU sensitive functions with no detection nor correction mechanisms. The others implement various mechanisms to detect a SEU and correct its impact on their SEU sensitive functions.

4.1.3 Step 3 - Verification and design Tools assessment

4.1.3.1 Tools

Theoretically, no specific tools other than standard tools used to implement and verify a PLD (synthesis, place and route, simulation tools) are required to implement a SoPC. Indeed, the microprocessor IPs should be sufficiently documented to be manually instantiated in a HDL code like any other IP or HDL component.

Nevertheless, some of the selected IPs can be parameterized to adapt its functionalities to the user needs. In some cases, those parameters can be generic parameters selected by the designer when instantiating the IPs in the SoPC HDL code. Sometimes, the parameterization is done through wizard tools that allow to graphically configure the IP. Such tools should be considered as a non qualifiable design tools according to the ED80/DO254 §11.4 definition. Then, referring to ED80/DO254 Figure 11-1, if an applicant intends to use a configuration wizard,the applicant would have to demonstrate a relevant service history (which may be difficult to get for recent technologies), or to independently review the output of the wizard. For some IPs, the latter is impossible since the wizard produces an encrypted HDL code or a synthesized netlist.

In the same way, the 3 main PLD vendors propose dedicated tools able to build and configure a complete SoPC, from a graphical interface, without writing a single HDL code line. Some of those tools produce a readable HDL code that can be manually reviewed, but the output of some others can not be independently assessed. Some PLD vendors impose the use of such a tool to be able to implement their microprocessor IPs. This approach may constitute a clear non conformity with the ED80/DO254 recommendations.

No specific tool is required to develop the software of the selected microprocessor IPs. Indeed, several compilers and linkers are available on the market and the software can be developed according to the ED12B/DO178B recommendations, like any software for a classical microprocessor COTS. Nevertheless, some concerns may arise regarding the verification of the software in the target computer environment (cf. §4.1.3.2 On chip debug facilities).

4.1.3.2 On chip debug facilities

A debug module is embedded within all the selected microprocessor IPs (for some IPs, the debug module is an option that can be implemented or not).

The debug module is likely to be used for software and/or hardware verification activities as required per ED12B/DO178B and ED80/DO254. In this context, unknown failure modes of this module may lead to the non-detection of a failure during the verification activities.

Thus, if the debug module is compulsory for verification activities, the designer should perform specific activities to gain some credit on this module.

4.1.3.3 Step 3 conclusion

In practice, the tools involved in the design of a SoPC should not have a safety impact on the application. Nevertheless, some of the selected PLD providers impose specific design tools. So, when selecting a SoPC solution and in addition to technical aspects, an applicant should take in consideration the required tool that may lead to not meeting the ED80/DO254 recommendations.

In the same way, since debug modules are subject to potential failure modes, the potential failure modes of the debug module should be taken into account when defining the hardware and software verification strategy.

4.1.4 Step 4 - SoC qualification assessment

4.1.4.1 Track record of production

This section applies only to hardware IPs embedded in the PLD silicon and addresses the entire PLD since it is not possible to assess separately the IPs.

The concerned PLDs top marking allows to track the various production parameters of the chip. Indeed, beside the reference of the circuit, it can be found on the chip:

- A code for the circuit revision,
- A code for the origin of the wafer,
- A code for the geometry of the circuit (minimum transistor dimension),
- A code for the date the device was assembled (week number and year),
- A code that identifies the lot number.

Then it is possible to track the device production process from the chip top marking.

4.1.4.2 Quality procedures

Generally, the public data addresses the quality aspects at chip level. They mainly deal with the processes and procedures applicable to the design, the production, the configuration management and problem solving of the silicon devices. They refer to international quality and qualification standards, and in some cases, an internal quality manual is published.

Nevertheless, they offer low or no visibility at all on the processes and procedures applied to the IPs. Thus, it is difficult to determine if the IPs have been designed according to a structured process, if they are under configuration management, if their problems are formally tracked and reported.

4.1.4.3 SoC qualification

This section applies only to hardware IPs embedded in the PLD silicon and addresses the entire PLD.

> One of the SoPC providers surveyed releases four times a year a “Device reliability Report”. This document addresses on one hand the finished product reliability that is measured periodically. On the other hand, it addresses the qualification of new devices, new wafer processes and new packages. Tests procedures, parameters and results are recorded in this document which recalls device per device the quantity of tested devices, the total amount of test hours, the number of failures and the resulting failure rate.

The reliability stress tests are conducted according to the conditions specified in JEDEC Solid State Technology Association’ s reliability test methods for packaged devices, JESD22.

Moreover, this PLD provider has been audited in 2007 and declared compliant with the requirements of the STACK international.

4.1.4.4 SoC service experience

Generally, no public data that could be used to claim some service experience has been found. In any case, a pertinent data collection that would allow to determine the number of applications, the number of IPs, the amount of operating hours, seems to be difficult to achieve.

Indeed, hardware IPs embedded in the PLD silicon are not necessarily used in the final application and a service experience based on the device itself would not be necessarily relevant regarding the microprocessor core.

In the same way, IPs providers can not have the visibility on the actual use of their IPs and even more when the IP is included in the library of a design tool.

Moreover, an error detected by an IP user is not systematically reported to the IP provider.

Finally, most of the IPs can be customized or parameterized, and the physical implementation of firmware and software IP is different for each application. Thus, the relevance of a service experience would be difficult to demonstrate.

As a consequence, an applicant would have to rely on its own experience to claim for a service experience.

4.1.4.5 Change process and problem reporting

Generally, IP customers are not notified when a new version of an IP is available or when a new problem has been identified. Customer notification processes usually address silicon errata, production processes, shipping procedures, etc…but do not address the IPs.

Most of the selected providers propose on their website knowledge databases which can be applications notes, FAQ, forums. These services can help the designer during the IP implementation. Nevertheless, no formal process has been found that would ensure that the published notifications are exhaustive and that all modifications or problems are reported.

Moreover, IPs are generally embedded in an IP library which is periodically completed with new IPs or adapted to new PLD families. PLD providers take advantage of a library update to correct potential errors of its IPs. The reason of the changes and the impact of the modifications are not formally notified to the customers. At best, each IP comes with a release note that identifies the differences from the previous version.

4.1.4.6 Obsolescence - guarantee - debug support

An IP is not subject to obsolescence as long as its data required to implement it (netlist or source code) are under configuration management.

All of the selected providers propose technical support to their registered customers.

4.1.4.7 Step 4 conclusion

The public data gives some information on the quality and qualification processes applied to silicon items, but offer no visibility on the design procedures nor the quality process applied to the IPs design.

Moreover, even if various means are proposed by the IPs providers to support and inform their customers, it seems that the detection, the collection, the reporting and the correction of bugs are not necessarily subject to formal processes.

4.1.5 Peripheral IP

In addition to the processor core, a SoPC is composed of peripheral IPs which extends the communication and/or the processing capacity of the processor core. These peripherals IPs can be:

- Designed by the SoPC designer,
- Given or sold by the processor core provider,
- Given or sold by any other IP provider.

The main processor cores providers propose peripheral IP libraries dedicated to their processor. Moreover, they are often embedded in the development tool required to implement and configure the processor IP.

The purpose of this section is to present the peripheral IP libraries proposed by the three main SoPC providers, and to assess their safety impact when embedded in safety critical applications.

Note: The homemade peripheral IPs are not addressed here since they can be developed according to the ED80/DO254 recommendations and then should not constitute a safety issue.

Note: The platform buses of the SoPC are specific peripheral IPs that offer on chip internal communication means.

4.1.5.1 General overview

The principle is almost the same for the three main SoPC providers. The XPS (for Xilinx), the SoPC builder (for Altera) and the CoreConsole (for Actel) are tools designed to help the user in designing a SoPC.

All those tools embed a catalog of peripheral IPs optimized for the vendor PLD devices characteristics, and designed to be easily connected to a processor core, distributed by the same vendor.

The peripheral IPs are classified according to their functionalities. Thus, the following main categories can generally be found:

- Bus and Bridge (on chip or external communication bus controllers),
- Clock, Reset (clock, reset, timers, watchdog management),
- Communication peripherals (USB, Ethernet, CAN, PCI, PCIe, UART, …),
- Memories (on chip memories),
- Memory controllers (controllers for on chip an external memories),
- Processing and Co-processing (FPU, filters, signal processing functions).

Each catalogue contains a large number of peripheral IPs. The catalogue is generally updated with each release of the embedding tool.

Building a SoPC from all three tools follows the same process. Once the processor core has been configured through a graphical wizard, the designer can build the SoPC following the actions below:

- Selection of the desired peripheral IP in the catalogue,
- Configuration of the IP through wizards specific to each IP,
- Connection between the peripheral IP and between the processor core through a graphical interface.

According to the tool, and the selected options, the outputs of such a process can be:

- A HDL file that instantiates all the project IPs, and establishes the connections between them,
- A netlist that can be used as a “mega IP” and inserted in a bigger design,
- A bitstream that can downloaded directly into the PLD device.

Note: The use of such a tool does not imply to select only peripheral IPs of its catalogue. The designer can also import homemade IPs or IPs provided by a third party. In any case, even if the IPs of the three main PLD providers are addressed here, the following considerations apply whatever their origin.

4.1.5.2 Safety impact assessment

All the peripheral IPs available in the three catalogues are generally provided with a data sheet which gives:

- The main features of the IP,
- The functional description,
- The IO signals and their behavior,
- The parameters,
- The performances and the resources according to the PLD target.

This data is generally sufficient to ensure a good implementation of the IP in the SoPC. They should be captured as hardware and software requirements that can be verified as part of the global PLD development process.

As for the processor core, these three kinds of peripheral IPs can be found:

- The software IP delivered with the source code,
- The firmware IP provided in a netlist or an encrypted HDL form,
- The hardware IP that is embedded in the PLD target silicon.

Each kind of IP does not present the same disadvantages and the same safety risks.

- Hardware peripheral IPs

The hardware peripheral IPs offer less visibility on their design and less possibility for customization. The designer will have to address them as black boxes which may be challenging for safety critical applications.

Indeed, the data sheets provided are implementation and user manuals, which do not necessarily address all the IP internal features or implementation tricks. Thus, due to the lack of documentation, it may be impossible to assess the compatibility of the IP with the project safety objectives. As a consequence, the designer would have to rely on one or several of the following activities:

  - To preclude or mitigate the impact of the IP on the rest of the design (by isolation or monitoring mechanisms),
  - To collect and justify relevant service experience data,
  - To perform advance testing to fully characterize the IP.

Note: Until recent years, hardware peripheral IPs were limited to quite simple functions such as on-chip memory or multiplier/accumulator. Then it was possible to achieve an exhaustive characterization. Recent PLD have now introduced much more complex hardware IPs, such as PCIe cores, which are more difficult to test exhaustively.

- Firmware IPs

The firmware IPs offer the same visibility than the hardware IPs, and thus may constitute the same safety risk. As a consequence, the above additional activities should be performed to ensure that the IP will not jeopardize the overall safety of the SoPC. Nevertheless, service experience may be practically impossible to demonstrate. Indeed, such IPs have often various tuning parameters that could help the designer to meet its functional requirements. Moreover, for each design that uses the same IP, the Place and Route tool provides a different physical implementation. For these reasons, it may be difficult to justify an identical context and to claim for some service experience credit.

- Software IP

Since the source code of the software IPs is available, it is possible to reverse engineer it and to rely on a design assurance process. Thus, the use of such an IP should not constitute a safety issue.

If the designer needs to implement a software IP as a black box, he/she will have to address it as a firmware IP.

- Tools

In the design flow described above, two kinds of design tools (according to the ED80/DO254 §11.4 definitions) must be considered: the wizards used to configure the peripheral IPs and the builder itself which generates the SoPC design.

The output of the configuration wizards should be manually assessed by the designer since those tools are not qualified and it may be impossible to rely on a relevant service experience. If the wizard produces an encrypted HDL module, there is no chance to assess its output. However, even if the source code of the IP is not available, the wizards often produce a "Wrapper HDL" file. The wrappers are plain text HDL files that instantiate the encrypted IP and configure its different generic parameters. It is then possible to manually review the wrapper files to ensure that the wizard has configured the IP in accordance with the user needs, and according to the content of the data sheet.

In the same way, as already stated, the SoPC building tools can produce a top level HDL file, a netlist or a bitstream. If the netlist or the bitstream is used by the designer, it must be assumed that the output of the design tool is not independently assessed and the designer will have to qualify the tool or to rely on service experience.

On the other hand, the automatically produced HDL file can be independently assessed (by review), and imported in a standard PLD development flow. In this case, the SoPC building tool will not constitute a safety risk.

4.1.6 SoPC conclusion

The analysis of the public data for the selected microprocessor IPs has highlighted the difficulties to achieve the steps 1, 2, 3, 4.

It has been seen that the processor IP cores have various complexity levels and present different issues considering their use in a safety critical application. The main issues identified are:

- A lack of visibility on the deactivation mechanism of unused functions,
- A lack of data allowing a safe implementation,
- A lack of safety mechanisms,
- A sensitivity to SEU/MBU,
- A lack of visibility on the design and data management processes,
- A lack of visibility on the error management and reporting processes,
- A lack of formal service experience records,
- An unqualified tool is required to configure the core.

In any case, on the basis of the public data, no processor core susceptible to be implemented as it is in a safety critical application has been identified (excepting the NIOS II SC whose design process can be relied on). Indeed, all of them integrate most of the above issues.

It has also been seen that it is impossible to rely on service experience or design data. Therefore a designer would have to mitigate or preclude the potential error modes of the processor core, which may also be challenging due to the lack of documentation.

As a consequence, the only solution may consist in getting support and/or additional information from the SoPC provider.

Moreover, the peripheral IPs may involve the same issues than processor cores. Nevertheless, they are often less complex, and the source codes are more and more available, which make additional activities easier to lead.

Except processor cores which require a specific unqualified tool to be configured, it has been seen that specific tools proposed by the PLD providers to built a SoPC do not constitute a safety risk in most of the cases, since their output can be independently assessed.

To summarize, it shall be considered that implementing a SoPC on the basis of the only public data does not allow to meet the ED80/DO254 recommendations and may constitute a risk for a safety critical application. In this context, two methods can be adopted.

When it is possible (source code available), the designer can reverse engineer the IP to release a complete ED80/DO254 life cycle data.

The other method would consist in defining an alternative design assurance strategy that would cover the lack of the public data and that would:

- Identify the IP functions that constitute a safety risk against the unexpected events of the final application,
- Define a domain usage where the IP safety impact would be limited,
- Define preclusion and mitigation methods at different levels to bound the impacts of unsafe functions.

This strategy will be addressed in section 6.

4.2 SOC MICROCONTROLLER

4.2.1 Step 1 - On chip cores identification and features determination

The results given in this section rely on the analysis of the public data available for the selected SoC. They are mainly:

- Data Sheets: Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations,
- Reference Manuals: These books provide details on individual implementations,
- Errata Manuals: Because some processors have follow-on parts, an Addendum is provided which describes the additional features and functionality changes. These Addenda are intended to be used with the corresponding reference or user manuals,
- Silicon errata: Identification of design fault,
- Quality handbooks which describe quality standards applied by SoC providers,
- Process Change Notification (PCN): The provider notifies their customers of changes that affect the form, fit, function or reliability of the products.

It must be noticed that:

- The consistency between the data and the silicon is not guaranteed as some data is released before the last silicon release. Thus, even in the case of a complete and precise documentation, a designer should consider potential outdated information in the documentation,
- Information usually given as public can be missing.

4.2.1.1 Features assessment – activation/deactivation

The complexity varies a lot from one SoC to another. Generally the constituents and the main features of the SoC are identified and explained in the data set given by SoC providers.

The configuration of the SoC is performed by specific programming configuration register. Thus, it is possible to deactivate or isolate specific functions and to configure this function by asserting bit of register.

The following main features can be found in the selected SoC. All the features can be configured in different ways according to the need of the final application.

Processor Core including the following key features:

- Several Arithmetic logic units and Floating point units,
- Vector processing unit:
  - Executes Single Instruction, Multiple Data (SIMD) operations. These operations allow for the execution of identical instructions to be parallelized across an array of data elements.
- Load/store unit:
  - Executes all load and store instructions as well as vector instructions and provides the data transfer interface between the general purpose registers, floating point registers, vector registers, and the cache/memory subsystem.

- Dispatch unit:
  - Unit which dispatch the instructions to the execution units.
- Register renaming:
  - Pipeline break could happen in case of instruction dependencies. To eliminate instruction dependencies, the micro-architectural algorithm assign architectural registers to different physical registers.
- Branch prediction
  - Unit process for guessing the direction or target of a branch.
- Memory management unit (MMU) allows to:
  - Give some Attributes to memory regions (Cachability, guarded access):
  - Access rights to memory regions (read only, write only, supervisor, users, executable/non executable),
  - Protect pages (pages belonging to a process are protected against access initiated by other processes),
  - Manage the swapping of page storages between mass storage and main memory (the number of pages can exceed the size of the main memory).
  - The last page translations are contained in the table called TLB (Translation Lookaside Buffer) within the cache.
- Instruction and data cache (2 levels):
  - Static RAM located within the processor which contains a copy of the most recent information read from external memory and their location,
  - Several cache controllers exist like fully associative, direct- mapped cache and the N way set associative,
  - Several types of algorithm replacements are supported.
- Dynamic power management:
  - Supplies or withholds power individually to execution units, based upon the contents of the instruction stream. The operation of DPM is transparent to software or any external hardware. For example, if no floating point instruction is being executed, the floating point unit is automatically clocked off.
- Dynamic frequency management:
  - Allows the ability to divide the processor to system bus ratio by 2 or 4 during normal functional operation in order to reduce the power consumption.

Infrastructure Core:

- Internal busses controller:
  - Provides a bridging structure for routing CPU and I/O-initiated transactions to target modules on the device,
  - Provides full cache coherency between the processor caches and the main memory (i.e. snooping),
  - Manages a complex arbitration logic between different masters and slaves connected on internal busses.
- Internal bridges:
  - Enables concurrent data flow between external Bus and SoC internal bus
- Address Translation and Mapping Unit:
  - Performs the translation between I/O mapping and the local mapping,

  - Defines how a transaction is routed through the device internal interconnection from the transaction source to its target.
- Memory controller:
  - Interfaces Memory with internal data bus of the SoC.
- Direct Memory Access controller:
  - Enables to transfer data between any of its I/O or memory ports or between two devices or locations on the same port.
- External bus controller:
  - PCI, PCI express, RapidIO, Ethernet...
- Programmable interrupt controller:
  - Enables the management of external interrupts, internal interrupts generated by I/O integrated peripherals and interrupts generated within the controller itself,
  - Servicing an interrupt involves; saving the context of the current process, completing the interrupt task, restoring the registers and the process context, and resuming the original process.
- Debug features:
  - Used to observe internal functions of the device,
  - Setting breakpoints in the normal program to generate specific interruptions,
  - Using a JTAG port to stop and control the microprocessor core. It is then possible to stuff an instruction in the normal program, to modify any internal register or memory, or to run the program instruction by instruction (stepping mode),
  - Tracing out the instruction stream executed by the processor to monitor in real time which the current instruction.
- Performance monitoring features:
  - Allow counting the occurrence of internal events within the SoC. For this purpose, the performance monitor contains several counters and associated control registers.
- Timers:
  - Generate time intervals as required by the software (Time events, Count events, Generate pulses, Interrupt the CPU).

Internal data flow:

The SoC microcontroller contains hardware to allow internal data transfer. The internal data transfer could be more or less complex:

- Internal data transfer is ensured by internal data/addresses buses implementing complex arbitration logic between different master and slave agents,
- The data transfer is limited to the Direct Memory Access operation which is responsible of all data movement between the on-chip cache memory, external memory, and the device peripherals. These data transfers include CPU-initiated and event-triggered transfers, master peripheral accesses, cache servicing and non-cacheable memory accesses, whose arbitration is configurable.

Register

SoC providers can allocate up to 1MB of register for SoC configuration, control and status. In fact only few parts of these registers are really used. A majority of the configuration, control and status registers are marked as 'reserved'. These reserved registers can be used by the SoC manufacturers for their own factory in-testing. It highlights that the SoCs can implement functions which are not documented in the public data. Furthermore, setting reserved registers can cause a machine check due to unspecified internal events.

The analysis of the SoC features highlighted the following issues:

- The release of some data comes before the silicon release. The documentation could be outdated.
- Some public data could be missing,
- Some SoC features could be incompatible with the application requirements. For example: The use of dynamic branch prediction, snooping, use of dual core, cache memories, MMU could be an issue according to the application, if an analysis is not performed to allow their use,
- Lack of information of the internal data flow (Internal safety mechanisms, arbitration algorithm, protocol...),
- The mechanisms behind the parameterization of activation/deactivation means are not always documented. Asserting specific bit to deactivate/activate functions could act as a reset, a clock disable or a logical disable,
- The number of configuration possibilities could be huge making the service experience analysis difficult for certification credit,
- The integration of multiple peripheral within the chip allows decreasing the power consumption of the application target. But the power dissipation is localized in a specific place of the board. The applicant could be tempted to use functions such as Dynamic Frequency Management, Dynamic Power Management without control,
- The analysis of the processor core has highlighted complex hardware features, which can be uncovered by classical software activities,
- Additionally to the documented features, there may be logics implemented on the SoC that have not been documented. These undocumented features may cause unpredictable executions,
- A large amount of memory is reserved for registers. However, if observed in detail, the registers mapping shows that the memory size actually used is much lower than the reserved size. It is possible that the reserved areas could be used for some shadow memory region for internal processing function.

4.2.1.2 Internal accessibility assessment

Both identified SoCs implement debug and performance monitoring functionalities. These functions give visibility into the SoC configuration, control and status registers, memory content, data transaction on I/O external interface but also limited visibility on data transfer on internal bus through a trace buffer.

This visibility into internal device operation is useful for debugging hardware, boot program and application software and reconstruction of the fetch stream.

To study the internal data flow of the debug information could help to determine which block or port originated a transaction, including the distinction between instructions and data fetches from the processor core. There is also possibility to obtain information such as transaction types, source

ID, and other attributes captured on the bus performance monitor facility, that can be used to monitor and record selected behaviors of the integrated device by counting specific events. The debug function and performance monitoring could be used as:

- A mean to model time of program execution,
- Monitor internal events during endurance and stress testing.

The debugger can help the designer to implement the SoC but does not offer the possibility to control and observe individually IP cores completely at hardware level. Some functions too deeply embedded in the SoC remain not accessible and can only be indirectly observed. It is the case for the internal communication busses, internal processor features, internal bridges…

4.2.1.3 User guides and Errata data assessment

Errata describing corrections or clarifications of the reference manual are provided for both SoCs. It is important to note that the release of the first reference errata can be made several years after putting the SoC on the market. These errata reveal incorrect or incomplete information about a feature described in the first release of the reference manual.

The misleading contents of the manual could lead the designer to have an unsafe implementation although he/she complies with the data provided by the SoC manufacturers.

Verification activities of the SoC may be performed to ensure the consistency between the SoC silicon and the data, and to cover issues raised by incomplete and incorrect public data information against the used revision of silicon product.

4.2.1.4 Silicon errata

The number of errata is directly proportional to the component integration and complexity. The errata list is becoming more important with the increase in density of transistors and the complexity of circuits. But it must be noticed that the number of errata decreases with the release of new revisions of silicon which fix identified bugs. Nevertheless, new silicon revision fixing bugs can generate new misbehavior or new limitations which were not implemented in the former silicon revision. For example, voltage de-rating could no longer be supported by new silicon revision.
For the majority of errata a description, a work around and potential dates of silicon correction disposition are provided. Some errata are not covered by a work around.
The analysis of errata highlights significant bugs, as shown by the following examples:

- I-cache parity errors can be reported incorrectly,
- Processor core does not behave as documented when a specific register is asserted,
- Data corruption possible for master writing to cache memory.

The capability of the SoC provider to give prompt public notification of errata is not guaranteed as the release of the first errata sheet could occur several years after the silicon release.

The analysis of errata highlighted the following issues:

- Evidence of lack of verification activities performed by the SoC providers before setting the chip on the market. It consolidates the fact that SoC microcontrollers are being targeted towards the high-demanding markets with quick turnover and new revisions, rather than correcting errors during verification phases performed before the release of new revisions,
- Evidence that the correction of bugs is also an opportunity of the SoC manufacturer to perform a set of modifications to improve the yield of the production. These news changes, if they are not analyzed, could be sources of additional bugs or new limitations,

- Evidence that there is no economic advantage for SoC microcontroller suppliers to collect and provide service history data and problems in a reliable manner,
- Bugs could lead to the system having an erratic behavior, if no work around is implemented or if the bug is not discovered before the entry into service.

It is essential that none of the design faults cause any failure associated with high criticality levels. The help of SoC manufacturers may be absolutely necessary to get available and last updated errata and work around in order to implement safely a SoC integrating a lot of hardware functionalities. Furthermore, it is essential to know what kind of modifications the SoC manufacturer performs between each silicon revision in order to detect potential risks of regression and to take credit on the design on previous silicon revision.

4.2.1.5 Step 1 conclusion

The constituents and the main features of the different cores are identified and explained in the reference manual of SoC providers. The configuration of the SoC is made by asserting specific registers enabling a large way of using the SoC for multiple applications. In most cases, the documentation may be sufficient to implement the SoC for non safety critical applications.

The following issues could represent a risk for safety critical applications:

- Missing or outdated information usually given as public,
- Internal processor and infrastructure core features (MMU, cache, cache coherency, internal data flow...) can not be deemed safe for flight and particularly for real time applications requiring determinism,
- The design assurance of the processor core cannot be supported by applicant application test cases developed as part of the ED12B/DO178B. These testing approaches may not achieve the desired structural coverage objectives required for safety-critical systems,
- Lack of completeness and correctness of reference manual against the used device,
- Lack of accessibility of some key internal structures,
- The lack of comprehensive knowledge of a SoC:
  - Lack of description of internal busses and switches, their configuration control and status register,
  - Undocumented function and undocumented register,
- No prompt publication of errata,
- The important list of errata for first silicon release,
- The consequence of SoC design fault within a safety critical system if this design fault is not well managed.

4.2.2 Step 2 - Fault tolerance and fail safe features assessment

4.2.2.1 Failure modes assessment

The provided data allows to identify functional blocks and their failure modes by trying to determine how the function might fail.

For example, for the SoC:

- For the MMU features within the processor, we could identify the following failure modes:
  - Incorrect translation,
  - Failed protection,
  - Access control failure,
  - Translation delay,
  - …
- For the external data bus:
  - Data transfer error,
  - Command error,
  - Babbling,
  - Packet transmission time out,
  - …

The SoC implements some fault tolerant support or fail safe mechanisms to ensure that the SoC can continue to correctly or safely function even when a failure occurs.

For example the SoC can implement:

- Fault tolerant features: Detection and correction of memories corruption like parity, and/or Error Correction Code (ECC) protection,
- Fail safe feature: Detection of internal data and address error by asserting specific status registers.

The safety mechanisms usually implemented in safety critical systems could be not taking into account by the SoC provider as such a mechanism could affect the performance and have no added value for the SoC target market.

Furthermore, even if safety mechanisms are implemented, it could be considered as insufficient for safety critical applications like:

- Incomplete protection against cache corruption,
- Incomplete detection of internal data/address corruption,
- No COM/MON architecture,
- No voting….

Thus for the majority of the functional failure mode, the SoC will have an unstable or hazardous state leading to a exception, reset or unintended behavior.

The major issues raised during the failure modes assessment are related to:

- The knowledge of the functional block to be confident that no significant failure modes have been forgotten,
- The analysis of the failure effects on the system due to lack of information concerning the input and output between each block,

- The credit which could be claimed by the SoC internal safety mechanism to detect and correct some failures.

4.2.2.2 Memory upset protection

SoCs implement mechanisms to detect and correct Single Event Upset and Multiple Event Upset. However, the coverage of detection and its correction is not complete and can be considered as insufficient according to the application:

- The Parity and ECC mechanisms on the cache and the main memories could not be covered by all types of memory upsets,
- No documented protection of the TLB (Translation Lookaside Buffer) of the MMU and, against SEU/MBU
- No documented protection of registers which are used for translation between the different memory domains (Inbound or Outbound),
- The branch prediction unit uses a BHT (Branch History Table) to draw up statistics on taken and not taken branches. The statistics are used to dynamically anticipate a specific branch. This BHT is most probably SEU/MBU sensitive (it is not documented). It is the same thing for the branch target instruction cache,
- Bus controllers often embed FIFO memories used to stall data when the bus is busy, or to move the data from one clock domain to another one. Depending on the technology used to implement those FIFOs, they may or may not be subject to SEU/MBU disruptions. Unfortunately, no information on potential embedded FIFO and on their technology has been found.

4.2.2.3 Internal Data transfers failure

The analysis of the internal bus topology has highlighted the following issues:

- Internal data/address flows are more or less complex according to the functionalities of the busses (multi-master, multi-slave, burst access…). The potential failure modes of communication busses are:
  - Modification of the data during the transaction,
  - Incorrect memory location access,
  - Never ending transaction, or too long transaction due to an arbitration problem or handshaking issue.
- There is no description of internal safety mechanisms to manage potential data transfer errors,
- A complete analysis of the bus involved in the critical safety path is not feasible due to the lack of information provided by the reference manual. The internal data flow management is a critical element, which will require the help of the component manufacturer,
- The SoC could contain internal switches among the PCIe/RapidIO/DMA interfaces fully autonomous with no possibility of configuration. Potential risk of on-chip bottlenecks may happen.

4.2.2.4 Unused functions deactivation

Several types of unused functions can be identified:

- Unused peripheral function like PCI interface, DMA…,
- Unused function, internal to the SoC or the processing core (that can not be considered as a peripheral) like: Cache, MMU, cache coherency module,
- Unused function deeply embedded in a used function like unused instruction, unused mode of a function (copy back for cache, dynamic power supply, parity used ...),
- Unused reserved or undocumented function like manufacturing functions, internal data flows, ...

The effort to understand the unused circuitry in order to see how it might affect the operation of the system, if those circuitries were activated, could be more or less difficult to achieve according to the type of the unused function.

In the public data provided, the mean to deactivate unused hardware function like PCI interface, DMA, DDR controller is well explained. For such a function, a specific register could be asserted according to the need or not to use these functions. The de-assertion of the register sets off the clock.

It is noticed by SoC manufacturers that an inadvertent activation of unused functions during operation without a hard reset could lead the system to a potential hazardous failure.

For other functions (debug feature, parity...), the action of the deactivation is not documented and it is impossible to know if it acts as a simple logical inhibitor, a software reset or a physical disconnect. An inadvertent activation of such function could be an issue as it is deactivated by a register set.

4.2.2.5 Timeout failure

The use of some SoC features could be an issue to guarantee a worst case execution time (WCET). Pipelining of instruction and branch, Cache memory (Instruction and Data cache L1, L2 Cache, Translation Look-Side Buffer…) make the WCET difficult to predict. Other sources of uncertain WCET computations are multimaster/arbitration for external busses and simultaneous use of multiple Direct Memory Access (DMA) engines.

4.2.2.6 Microcode design errors

The SoC can implement microcodes to execute specific functions. Microcode is a program stored within an internal read-only memory array to execute complex machine language instructions like a cache flush routine. Microcodes could also be used to handle special events such as exception, interruption or processor resets. A SoC provider does not provide this microcode nor its design process data.

This lack of visibility of the microcode and the knowledge of how it interferes with the operational code can be a source of errors.

4.2.2.7 Step 2 conclusion

The functional failure mode and effect analysis of the SoC and their effect could be difficult to reach if the applicants intended to use complex internal function interaction and if they have no additional information about the internal function interaction.

The SoC implements no or partial fault tolerance or fail safe support which could be considered very crucial for the correct execution of a SoC under all circumstances. The lack of fail safe or fault tolerant support implemented by the SoC could lead the system to an abnormal behavior in case of design errors leading to a failure.

The analysis of potential inadvertent activation of unused functions could be difficult to lead according to the type of unused functions which need to be considered.

Finally the use of some features (without precaution) could cause unpredictability of the run-time tasks.

To address all these issues, the following activities could be performed by the applicant:

- Access of private data is crucial at least to complete the internal failure mode and effect analysis,
- Limitation of usage domain: For example, to limit the impact of unpredictable run-time task, the designer could:
  - Deactivate the cache,
  - Use cache locking features if it is supported by the SoC,
  - Partition the cache to eliminate interferences between tasks,
  - Empty the cache upon a context switch,
  - …
- Implement safety net at system , hardware or software levels:
  - Dissimilar redundancy: This class of architecture could possibly reveal random physical failures and design errors. The design should assure the independence of requirements, algorithm, data, and other potential sources of design error,
  - Wrapping encapsulates and protects data as it passes through an I/O IP core. An example would be the use of a communication package whose robustness is unknown. Data to be transferred can be encrypted prior to the communication and decrypted after the data are transferred through the COTS component. This encryption could possibly include error detection and correction schemes to ensure data integrity,
  - Watchdog timer which could detect a delay of execution time in a process,
  - …

4.2.3 Step 3- Verification and design Tools assessment

The usage of tools to develop a system implementing a SoC is essential for hardware debug, software debug, hardware/software integration, testability….

4.2.3.1 Tools

SoC manufacturers offer classical software tools as compiler, linker, and analyzer to configure the SoC registers, to generate and test the source code.

No other tools are needed.

4.2.3.2 On chip debug facilities

The on-chip debugger logic is part of the actual SoC silicon. Usually the on-chip debugger provides the means to set simple breakpoints, to query the internal state of the chip, and to single step through code. The SoC may implement the following debug facilities,

- Core debug facilities:
  - Internal debug: A debugger is present in the EEPROM. This debug mode provides traces on each instruction and traces which branch is taken. This mode affects the CPU performance,
  - External debug: This mode corresponds to JTAG emulation. It is possible to set breakpoints, to trace the program and to view memory content. When the processor is running, the JTAG probe does not affect the performance. The e600 does not indicate the debug station nor which instructions are being executed,
  - Real time trace. This mode allows to track code executions, determining what branches have been taken, provided that instruction accesses are performed over the SDRAM bus.
- Infrastructure debug facilities:
  - It offers a watch-point monitor and traces buffer capabilities to provide some visibility of internal buses. The trace buffer can capture information on the internal processing of transactions to selected interfaces. ,
- Performance monitoring functionality can be used to monitor and record selected behaviors of the integrated device (DMA Events, DDR Memory Controller…).

On-Chip debug, on COTS processors, is usually activated and deactivated only through JTAG port connection and disconnection. JTAG port connection is detected by common electrical level on several activation pins. When JTAG is unplugged, levels are stuck at 0 or 1 by pull-up/down resistors.

4.2.3.3 Step 3 conclusion

All the debug modes are essential to debug the software and the hardware. An error of the tools will not directly affect the safety of a flight or cause a failure related with criticality levels A, B, C, or D, if testing on the physical target is performed. At least, an error of this debug facility or tools could lead to potential impacts in the verification activities if no other means are used to verify the system behavior. In this case it will be essential to assess the debug module if the whole verification activities rely on this module.

However the tools and debug facilities can significantly affect development feasibility as these facilities could be used to achieve a level of testability.

4.2.4 Step 4 - SoC qualification assessment

4.2.4.1 Track record of production

The concerned SoC top marking allows tracking the various production parameters of the chip. Indeed, beside the reference of the circuit, it can be found on the chip:

- A code for the circuit revision,
- A code for the origin of the wafer,
- A code for the geometry of the circuit (minimum transistor dimension),
- A code for the date the device was assembled (week number and year),
- A code that identifies the lot number.

Then it is possible to track the device production process from the chip top marking.

4.2.4.2 Quality procedures

Generally, the public data addresses the quality aspects of the SoC and the SoC manufacturer. They mainly deal with the processes and procedures applicable to the design, the production, the configuration management and the problem solving of the silicon devices. They refer to international quality and qualification standards (Six Sigma quality principles ISO-based quality standard...), and in some cases, a internal quality manual is published.

Nevertheless, they offer low or no visibility at all on the processes and procedures applied to the design and verification of the SoC. Thus, it is difficult to determine if the SoCs have been designed according to a process compatible with ED80/DO254 expectations.

4.2.4.3 SoC qualification

The qualification process is the method by which the SoC manufacturer confirms that the reliability of their design, processes, products and packages exceed the expectations of their targeted market. Series of mechanical, electrical and environmental tests according to the JEDEC procedures or MIL-PRF-38535 performed on the component are included in the quality handbook or other public data of the SoC provider. The analysis of these procedures may highlight the lack of testing against specific aeronautical constraints like radiation effects or EMI. The levels reached during these qualification tests are not published.

If a manufacturer has not considered testing certain environmental scenarios applicable for aircraft applications, then the SoC may submit additional testing.

Furthermore, the user manual and data sheet mentions that the bus controllers (PCI express, Rapid I/O, I2C...) are compliant with the dedicated standard. However, no formal proof has been found.

Potential results of inter-operability tests performed by independent entities are not provided.

4.2.4.4 SoC service experience

The SoC microcontrollers are initially designed to cover high volume markets such as the telecom, industrial, automotive or consumer markets. The public data does not provide information on the number of applications, number of design types, number of hours of operation which could be used as proof of in-use data to comply with ED80/DO254 §11.3 (product service experience). It must be noticed that a SoC, which is available for 2 years, has multiple customers and has been sold in millions.

The main issue is to take credit on this fact as it is impossible:

- To demonstrate a relevant service experience for the entire SoC due to the large possibility of configurations,
- To know the effectiveness of problem reporting.

Most SoCs are developed from pre-qualified IP blocks, implementing hardware functions defined by industry standards such as PCI express, Rapid IO... The hardware blocks are put together using tools in different SoC designs. Thus, it would be possible to take credit on the service experience of each IP implemented within different SoC families. To be relevant, this IP needs to be in the format of hardware IPs (i.e.: GDSII files). However, the public data does not provide any information about the similarity between different SoC families.

4.2.4.5 Change process and problem reporting

A SoC quality handbook describes a way to report problems using appropriate processes, to research and provide corrective actions for customers' product returns and complaints, as well as internal issues. In these processes, it is not mentioned if problems are reported for all final users.

The SoC providers' websites generally propose a section dedicated to Product Change Notices (PCN for all SoC products). The analysis of these web files could allow to determine the current content of the PCN and to determine if this content is enough to ensure safe implementation of a modified SoC.

The PCN may contain the following information:

- Affected change categories,
- Description of change,
- Reason of change,
- Impact of product (form, fit, function or reliability),
- Qualification status, qualification plan,
- Reliability data summary,
- Electrical characteristic summary,
- Changed part identification.

It must be noticed that a SoC manufacturer may reserve the right to make changes without notification when fit, form, function, quality and reliability are not affected.

The analysis of PCN highlights that the change reporting could be incomplete to ensure a safe implementation of a modified SoC device.

4.2.4.6 Obsolescence – guarantee – debug support

The process used to be notified of a SoC obsolescence is described in the public data or in the SoC providers' websites.

It is generally possible to place an order within 12 months after the obsolescence notification and to have an additional 6 months to take delivery of the ordered product.

A typical obsolescence record contains the following information:

- Detailed description,
- PCN tracking number,
- Last order date (12 months after notification),

- Last delivery date,
- Product identification (affected products),
- Identification of replacement product, if applicable.

SoCs are introduced to address a set of industrial markets. Therefore, several years of production and support may be ensured. On the other hand, the SoC provider can stop to produce SoCs if the business is no more profitable or if they decide to propose new products with higher performance and better yield. Thus, the technology roadmap of the SoC is difficult to predict without the support of the SoC manufacturer.

An early obsolescence could affect the objectives of service experience coupled with the problem reporting of multi-customer.

4.2.4.7 Step 4 conclusion

SoC microcontroller capability to produce SoC components, with a high quality and reliability level, is not an issue as they need to satisfy all customers, which buy millions of pieces.

The major issue is to get relevant service experience data which seems to be unachievable regarding the ED80/DO254 objectives and the number of configurations offered by the SoC.

The second major issue is the confidence level that can be given on:

- SoC manufacturers' responsiveness (response time when critical problems appear, answers to questions, requests for help for installing, testing, using products),
- SoC manufacturers' capability to provide adequate failure analysis and to provide timely response to failure during the aircraft life service.

4.2.5 SoC microcontroller conclusion

The public data has highlighted the difficulties to achieve the questionnaire defined in the methodology section.

Both selected SoC manufacturers do not provide detailed public data to satisfy the ED80/DO254 standard.

The strategy was also to assess the public data against the alternative methods and key attributes. The two selected chips have different levels of complexity, configurability, maturity, which impact the level of confidence of their design quality.

The list below summarizes the major issues:

- Missing information usually given as public,
- Internal processor core features (MMU, DPM...) can not be deemed safe for flight and particularly for real time applications requiring determinism,
- SoC features (cache coherency, switch engine, DMA transfer) can not be deemed safe for flight and particularly for real time application requiring determinism or application requiring portioning protection for specific memory access,
- The lack of consistence between the chip and the data has been highlighted by manual errata and silicon errata,
- The errata list is becoming important (due to the density of transistors and the complexity of the circuit),
- The number of configuration possibilities of the components makes it difficult to achieve relevant service experience. Multiple designs with the same SoC can be used completely differently,

- Software solution alone may not address the problem of the hardware functions implemented in the processor core,
- Core accessibility (no pin access for testing the SoC internal hardware feature),
- The management of potential unused functionalities,
- The lack of knowledge of internal data flow, internal bridging makes it difficult to perform a safety analysis,
- Mitigation of memory upset (ECC, parity) is not implemented or not completely implemented,
- No service experience records,
- Problem report processes and change management processes not in line with usual aerospace requirements.

The technological trend will enforce the current identified issues:

- 65nm, 45nm, 35nm recently introduced or coming soon onto the market lead to more complex devices:
  - Testability and visibility will decrease,
  - More and more internal data transactions.
- Developing software before real physical hardware is available:
  - Co-design concept imposed by commercial market driving to earlier obsolescence.
- Multi-core design required by clock frequency limitation (single-core processors might reach the point of performance stagnation):
  - Need to implement new real time scheduling algorithms,
  - Software debug will be difficult on final target,
  - Need to use modeling for software debug.
- Reduction of data transfer latency:
  - More concurrent data transfers within the chip with multiple on-chip switching,
  - Sensitive signals will need to be utilized in safety critical design. Communication peripherals like PCIe, SRIO and GigE that run at several GHz are becoming the norm.

5. PRIVATE DATA ASSESSMENT

Positive verbal commitments have been received by the majority of SoC providers to consider these issues in their business. But after a set of discussion with the SoC providers, the access to private data and their assessment were considered not relevant and useless for the survey. Indeed, the result of the private data assessment would have no impact on the main goal of the survey, which is to propose amendment to current certification process. This assessment would just give information on the selected item and could not in any case provide general conclusions on SoC providers.

Moreover, it seems more important to assess the possibility to involve the SoC providers in the certification process. Indeed, we have seen that the SoC certification may be difficult to reach without an access to private data and thus without the cooperation of the SoC providers. Thus, it is important to determine to what extent they can and they want to be involved in the certification process by giving access to their private data.

The following sections give the position of the main SoC providers and highlight their strategy to address the aeronautic market. That information is issued from the various informal exchanges that we had with the SoC providers.

5.1 SOPC PROVIDER POSITION

PLD manufacturers, and more generally IP providers, are considering the above concerns and are aware that the aeronautical market needs new solutions to be able to beneficiate of the last technological novelties. They are defining a positioning strategy and should propose in the following months solutions in line with the ED80/DO254 recommendations, such as qualifiable automatic tools and libraries of IPs developed according to ED80/DO254. Some PLD vendors already propose such IPs.

5.2 SOC MICROCONTROLLER POSITION

The SoC microcontroller business is driven by high volume markets to compensate important investment required to propose last state of the art technologies. It means that there is still no interest for them to propose products for the aircraft industry only.

SoC providers may agree to share some private data on the basis that they gain in turnover and corporate image from the use of their components within safety critical systems. It means that if one SoC user can have an access to private data with NDA in order to complete the questionnaire, it is not obvious that all SoC users can have this privilege. On the other hand, SoC providers could prohibit the use of their component in safety-critical applications unless there is a formal agreement between the parties governing such a use. SoC providers consider that it is the liability of the applicant to use design techniques to mitigate the effects of potential failures.

Moreover, SoC manufacturers could be reluctant to sign a NDA and share data with a public agency like the EASA (or FAA). This could be an issue for the SoC users to provide information to Airworthiness authorities to achieve a certification as the data may not be assessed by the EASA or the FAA.

SoC providers would prefer getting an accreditation from major aircraft manufacturers to allow the use of their components on the basis of their general design process assessment.

In this context, it may be impossible to meet Certification Specifications requirements in the frame of the current certification process. Another approach should be proposed.

6. RECOMMENDATIONS

The following list of recommended activities can provide support to both regulatory agencies and industry.

This section starts with SoC microcontrollers because the recommendations for the SoPC rely partially on what is proposed for the COTS.

6.1 SOC MICROCONTROLLER RECOMMENDATIONS

6.1.1 Introduction

Current certification practices may need to be increased with additional and refined requirements to cover the evolving system on chip technologies. The current certification practices defined by the compliance against ED80/DO254 §11.2 (Electronic component management process) and §11.3 (Product service experience) can be considered only as a beginning.

The methodology of certification should take into consideration the conclusions from the survey made on the public and private SoC data assessment:

- The SoC microcontroller may misbehave and may impact the safety by unexpected behavior due to resident hardware design errors,
- The SoC microcontroller public data may contain inconsistencies against the delivered devices,
- Achieving certification support from SoC microcontroller manufacturers for aircraft requirements on safety, security and certification is not reachable due to NDA issues,
- The selection of SoC microcontrollers is driven by latest novelties in term of performance and integration. It means that relevant service experience in the aeronautical field does not seem reachable,
- There is no economical advantage for SoC microcontroller suppliers to collect and provide service history data and problems in a reliable manner,
- SoC microcontrollers are being targeted towards the high-demanding market with quick turnover and new revisions, rather than correcting errors during verification phases performed before the release of new revisions.

The SoC microcontroller certification aspects cannot be limited to component level aspects, but need to include the system, software and hardware levels as well as safety aspects.

Figure 6 recalls the ED80/DO254 §2 contents which illustrates relationships between the system development process for airborne systems and equipment, the safety assessment, the hardware and software development processes. The SoC should be assessed within the overlapped Safety/Hardware/Software domains.

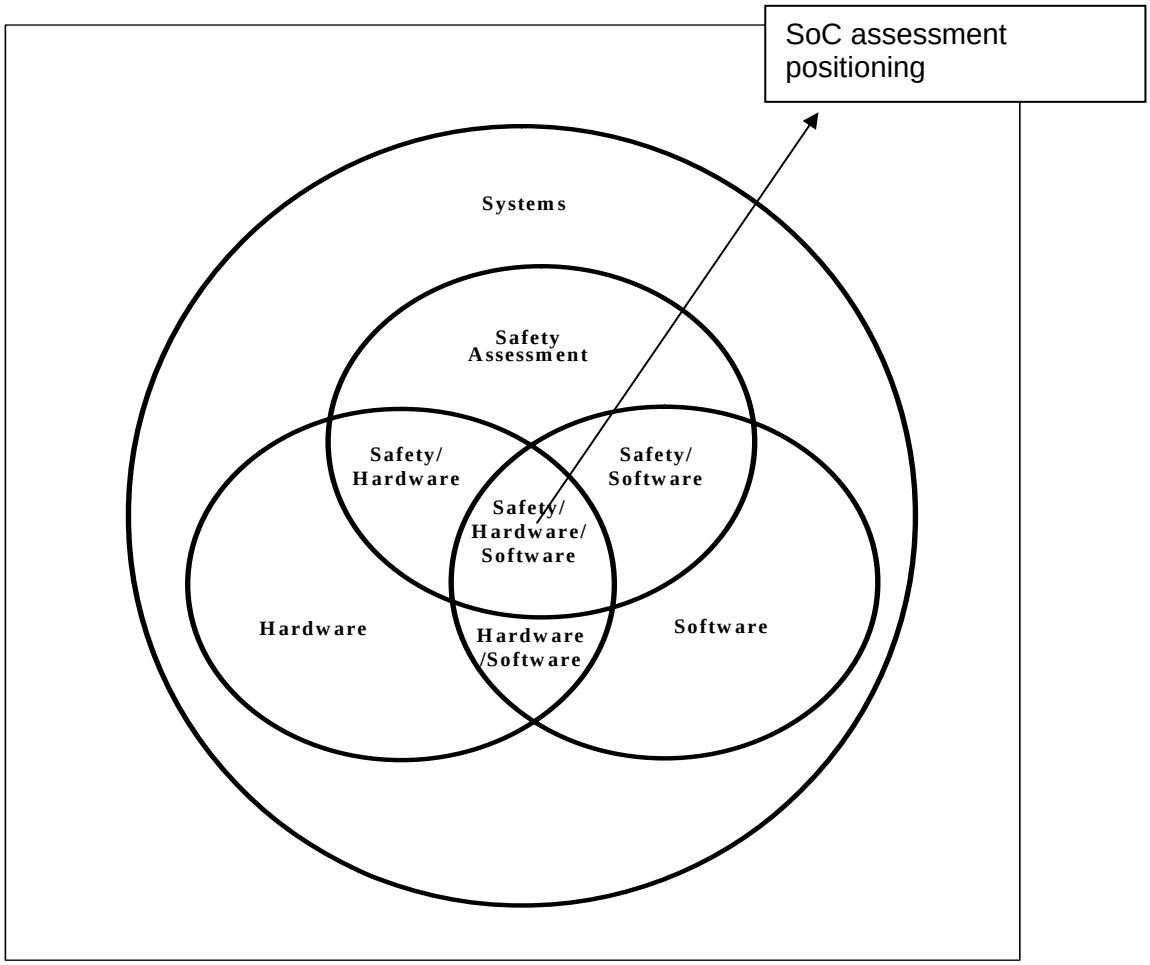


Figure 6 - Relationships between Airborne Systems, Safety Assessment, Hardware and Software Processes

Furthermore the ED80/DO254 §11.2 mentions that '*the use of COTS components will be verified through the overall design process*'. The intent of this statement could be that the hardware design assurance process and associated certification activities need to be ensured at higher level than device level. It should be considered at higher level: circuit board assembly and Line Replaceable Unit (LRU) levels.
As a result, it seems inconsistent in the FAA approach (AC 20-152) and the current European hardware certification practices to invoke ED80/DO254 only at device level.

The cooperative assistance of SoC microcontroller suppliers could be essential and should go beyond the normal commercial component supplier interactions with the customer, especially if the applicant intends to reach certification for a level A or B design. This cooperative assistance needs to be assessed by the certification applicant and Certification Authorities with other means than intrusive data assessment due to confidentiality concerns. These means are described in §6.1.2.

6.1.2 Approach overview

Figure 7 flow chart indicates the SoC microcontroller usage assessment considerations and intended activities when a SoC microcontroller implementation is planned within a safety critical system.

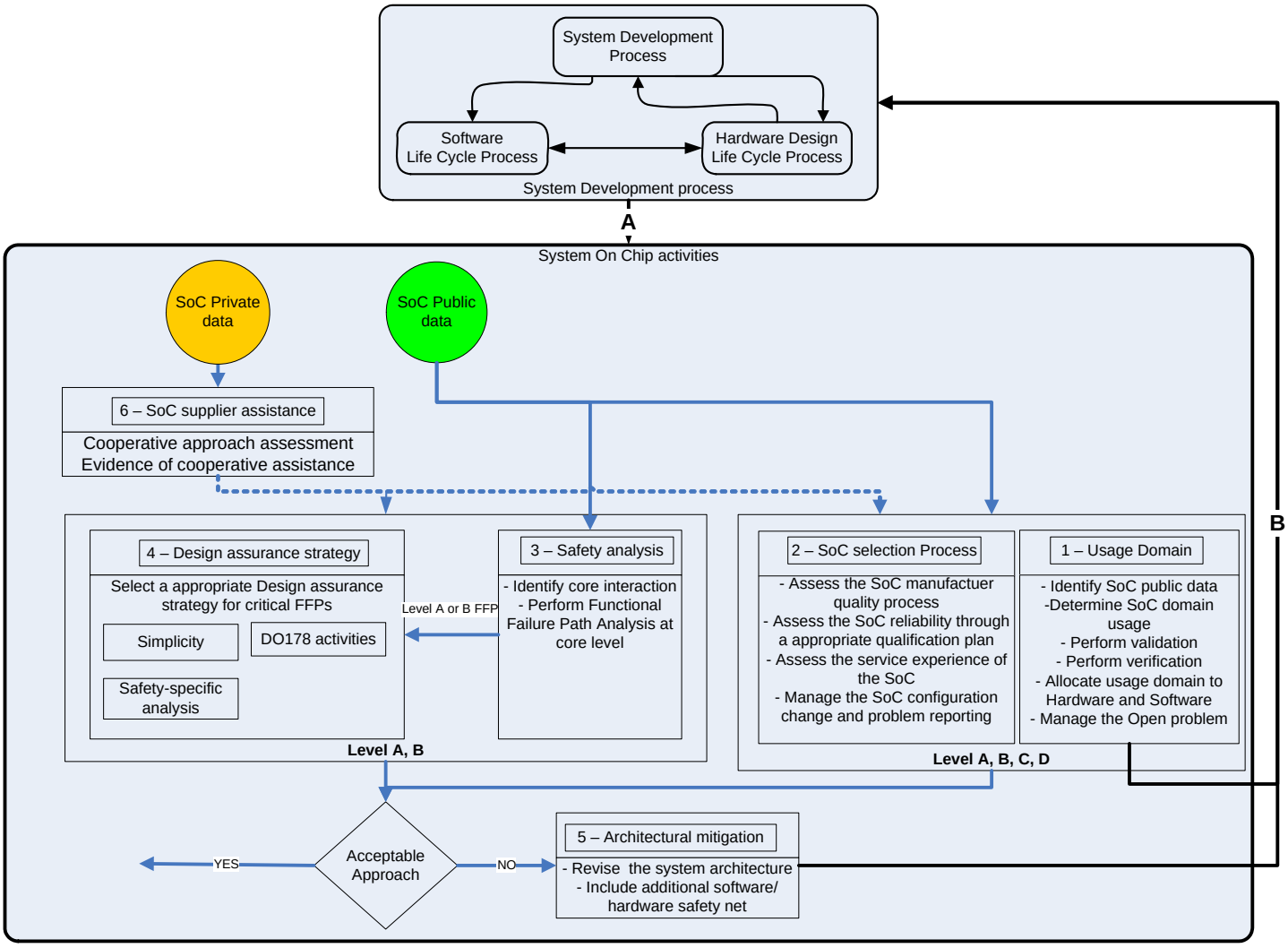


Figure 7 – Recommendations for SoC microcontroller usage

The SoC microcontroller usage needs to be assessed through:

- System development activities compliant with ED80/DO254, ED12B/DO178B, ED79/ARP4754, ARP4761,
- System on chip activities based on a black box analysis applicable for level A, B, C, D plus a grey box analysis applicable only for level A, B.
    - Black box:
        - Usage domain definition [1]: The objective is to define a usage domain of the SoC based on the SoC data and the requirements defined by the system development process. The usage domain should be validated and verified.

- SoC selection process [2]: The SoC and its manufacturer shall be assessed against a component management plan. A reasonable service experience should be demonstrated with an appropriate change management and problem reporting process implemented.

o Grey box:

- Safety analysis [3]: The objectives of the SoC safety analysis is to determine Functional Failure Paths (FFPs) which contribute to catastrophic or hazardous system failure conditions,
- Design assurance strategies [4]: For level A and B FFP, a design assurance strategy should be selected. The most suitable design assurance method may vary depending of the different functional paths within the SoC. The recommendation proposes three design assurance strategies: ED12B/DO178B activities, safety-specific analysis and simplicity.

If the approach is not reachable, the applicant shall:

- Modify the system architecture and/or implement additional hardware or software safety nets [5],
- Abandon the use of SoC microcontrollers within the application.

6.1.3 System development activities and SoC activities interaction

The system, software and hardware assurance processes and safety assessment need to be implemented according to their current standard for certification. The SoC microcontroller assessment should also be in relation and interaction with the hardware (SRU, LRM), software, system processes and safety assessment. These overlaps illustrate the need for a coordinated interaction between the processes to ensure that the assurance requirements of the system function are satisfied when the applicant intends to implement a SoC within a safety critical application. A flow of information between the life cycle processes and the SoC should be implemented to ensure that the SoC implementation will satisfy the system safety, functional and performance requirements.

The information flow from system, hardware and software development processes to SoC activities is represented by the letter A in the Figure 7.

This information flow may include:

- Design (function, performance) and safety requirements allocated to the SoC,
- Design assurance level for each function, along with its associated requirements and failure conditions, if applicable,
- Installation, ergonomic and environmental requirements allocated to the hardware,
- …

The information flow from SoC activities to system, hardware and software development processes is represented by the letter B in the Figure 7.

This information flow may include:

- Derived requirements needed for hardware/software integration, (e.g. Definition of configuration register, address mapping…),

- Integration of the problem reports work around that may have an impact on hardware or/and software requirements,
- Implementation architecture, including fault containment and fault mitigation strategies,
- Implementation of safety net (wrappers, CRC) impacting the hardware/software design,
- Failure effects of potential anomalous behavior of internal SoC functions,
- Latency analysis data relevant to system requirements (e.g. hardware provisions for fault monitoring, fault detection intervals and undetectable faults).

6.1.4 SoC microcontroller activities

The detailed explanation is given within the following table split into 4 columns:

- Column 1: Objectives,
- Column 2: Reference,
- Column 3: Activities which should be performed to comply with objectives,
- Column 4: Rational column justifies the needed activities. It refers to the public and private data assessment conclusions.

1- Usage Domain			
Objectives	Reference	Activities	Rational – Output of public/private data
SoC microcontroller usage Domain definition	ED80/DO254 § 11.2.2 item 1 ED80/DO254 §11.2.1 Item 6	The applicant should identify SoC microcontroller Public data • Identify at least the following data (users manual, data sheet, application note, device and users manual errata), • Compare the release date of the data with the release of the silicon. The applicant should determine a SoC microcontroller usage domain for the intended application. The SoC microcontroller usage domain should determine: • A usage domain of processing core, • A usage domain of the infrastructure. The usage domain should identify: • Used functions configuration: o Function description, o Mode of operation, o Fault tolerance internal mechanisms (ECC, parity) configuration, o Relation with the configuration registers. • Internal data path configuration: o Identification of data flow, o Master/slave agents,	Step 1 Availability of information for the SoC microcontroller is very significant for understanding the features, including the current design fault, the qualification data, the quality data and also the configuration management information. The comparison of the release date of the public documentation against the release date of the corresponding SoC microcontroller helps to detect potential outdated data. SoC implement a set of functions which could be used in different ways. The usage domain will allow to identify the function as well as its configuration. Example: MMU usage domain: Effective address to physical address translation could be performed according to 3 modes (Real mode, block or page mode). All these modes are supported by the MMU. The usage domain should contain which mode will be implemented. An unused hardware circuitry could be comparable to unused code in software application. In this case the designer of the system using the SoC microcontroller should make an effort to understand the unused circuitry in order to see how it might affect the operation

1- Usage Domain			
Objectives	Reference	Activities	Rational – Output of public/private data
		o Logic arbitration and protocol, o DMA transfer configuration, o Cache coherency configuration, o Relation with the configuration registers. • System configuration: o Address translation and mapping between : ▪ External peripheral and local mapping, ▪ Local master and local slaves. o Reset management and power on configuration, o Clocking configuration: Identification of the different clock domain, o External and internal interrupt management, o Relation with the system configuration registers. • Unused function: o Means of deactivation implemented by the SoC, o Identification of external means to control inadvertent activation of unused function • Work around requirements required by errata sheet, • Interface and implementation requirements including signal description	of the system if those circuitries were activated. Several types of unused functions can be identified: • Unused peripheral functions like PCI interface, DMA… • Unused functions, internal to the SoC or the processing core (that can not be considered as a peripheral) like: Cache, MMU, cache coherency module, • Unused functions deeply embedded in a used function like unused instructions, unused modes of a function (copy back for cache, dynamic power supply, parity used .....) • Unused reserved or undocumented functions like manufacturing functions, internal data lows, ... Traceability shall allow to link configuration registers and the usage domain. It will help the system designer to ensure that the users manual, errata, application notes have been well analyzed and taken into account. Traceability will help system designers for their impact analysis in case of the users manual modification.

1- Usage Domain			
Objectives	Reference	Activities	Rational – Output of public/private data
Validation of the usage domain	ED80/DO254 § 6.3.3.1 § 6.3.3.2	The applicant should validate the usage domain with respect to safety and system specifications: • The use of features shall be justified and consistent with system, hardware, software and safety requirements, • The deactivation of unused function(s) shall be justified and consistent with safety requirements. The applicant should demonstrate the consistency between the usage domain, the SoC data (users manual, errata, application note and usage domain) and the system requirements. • Consistency should be ensured by traceability means between SoC data and usage domain, • Consistency should be ensured between SoC usage domain and the system, software and hardware requirements.	Step 1 & Step 2 The usage domain of the SoC microcontroller should be consistent with the system, hardware, software and safety requirements. Example: • Consistency with performance requirements, • Consistency with qualitative safety objectives, • Consistency against real time application. For example to comply with WCET the cache functionalities could be unused or used by taking some precaution, • Consistency against Environmental constraints (EMC, Thermal, Radiation, mechanical stress, power...).

1- Usage Domain			
Objectives	Reference	Activities	Rational – Output of public/private data
Verification of the usage domain	ED80/DO254 § 6.2.2 §6.3	The applicant should verify the usage domain • Debug function and the performance monitoring function should be analyzed in order to determine the level and type of monitoring of internal features. • The validity of the usage domain should be ensured by a set of verification activities mainly based on: o Test (used functions, fault tolerant support verification, unused function deactivation effectiveness, errata work around verification) o Analysis as ▪ Design margin analysis to verify that the SoC implementation takes into account potential variability of component parameter, ▪ Similarity analysis to compare characteristics and usage of SoC functions previously certified, ▪ Impact analysis of inadvertent activation of unused functions, ▪ Single Event Upset and Multiple Event upset analysis. • Consistency between usage domains, verification procedures and results should be established.	Step 3 Availability of debuggers will facilitate software/hardware development and debugging, and can significantly affect development feasibility due to the capability to observe internal functionalities and events Step 1 & 2 The verification activities will mainly allow to make sure that the available sources of information are valid for the applicant's application. The testing activities should be performed to ensure that used functions, fault tolerance mechanisms, the unused functions deactivation and the errata work arounds produce the expected executions. The testing activities could rely on verification performed during software/hardware integration at board level. It should be considered that SoC microcontroller environmental qualification process (refer to SoC selection process) allows to give a verification credit on physical characteristics of the SoC. Inadvertent activation of unused functions should be analyzed in order to demonstrate that the unused function is isolated (clock off) or to demonstrate that there is no potential anomalous behavior that could have an adverse effect on safety.

1- Usage Domain			
Objectives	Reference	Activities	Rational – Output of public/private data
Allocation of the usage domain to Software and Hardware	ED80/DO254 § 2.1.1	The applicant should feedback the usage domain to the requirements capture process of the system, hardware and software development according to an appropriate allocation analysis. For example: • The configuration of the SoC will be allocated to the software process, • The physical implementation (place and route constraints, power on configuration.) will be allocated to the hardware process, • Safety mechanism, architectural mitigation, errata work around will be allocated to system, hardware or/and software processes.	Step 1 & 2 Ensure a good allocation of the usage domain to software and hardware design processes which shall follow the ED12B/DO178B and ED80/DO254.
Open Problem management	ED80/DO254 § 7.2.3	The applicant should assess all open problems (Errata) against potential adverse effects on the system. • The assessment should be performed on: o Problems coming from SoC microcontroller suppliers, o Applicant's application service experience. • The applicant may also have to establish limitations on the use of certain features that highlight history of design problems or errors, • Open problems which could have safety impacts should be closed or managed with a work around.	Step 4: Some design errors may have work arounds and might not affect the correct execution of a SoC microcontroller, if a particular configuration is provided. Some other errors may not have work arounds, and they may consistently affect the correct execution of a SoC microcontroller. It is essential that none of the design error causes any failures associated with high criticality levels (Level A, B, or C).

2 – SoC microcontroller selection process			
Objectives	Reference	Activities	Rational – Output of public/private data
SoC microcontroller Manufacturer's quality process	ED80/DO254 §11.2.1 Item 1, 2, 5	The applicant should assess the SoC microcontroller manufacturer's quality process: • SoC manufacturer should have a documented quality management system including the o SoC microcontroller manufacturer's quality assurance procedures, o SoC microcontroller quality control system (Screening, yield…) • The SoC manufacturer should demonstrate the repeatability of the SoC microcontroller production	Step 4: This activity is part of the application of component management plan as defined by the ED80/DO254 §11.2. or other international standards like IEC62239.
SoC microcontroller reliability assessment	ED80/DO254 §11.2.1 item 5	The applicant should assess the SoC microcontroller environmental qualification process which establishes component reliability. • The applicant should define an environmental qualification plan, test procedures, sampling and acceptance criteria (with the defined margins) regarding the final application. • In case the environmental qualification is executed by the SoC microcontroller manufacturer, the applicant should review the defined qualification testing. In case stress levels in the SoC microcontroller qualification plan do not equate or are below what is required by the end application, additional testing should be defined.	Step 4: If a SoC microcontroller manufacturer has not considered testing certain environmental scenarios, then the SoC microcontroller may be unsuitable for avionic products. Testing the SoC microcontroller against radiation effects that arise in avionic environments is an example of such a situation. This activity is part of the application of a component management plan as defined by the ED80/DO254 §11.2. or other international standards like IEC62239.

2 – SoC microcontroller selection process			
Objectives	Reference	Activities	Rational – Output of public/private data
Service experience analysis	ED80/DO254 §11.3	The applicant should demonstrate that the SoC microcontroller has a satisfactory wide diffusion with several million hours of estimated operation. The following criteria could be used: • SoC microcontroller service experience environment o Market target of the SoC microcontroller (industrial and automotive market will be preferred as consumer market references). • SoC microcontroller stability and maturity: o Date of the first release (at least the SoC microcontroller shall be in service for 2 years), o Evolution of silicon revision and errata. The nature of evolutions shall be analyzed to demonstrate that the SoC microcontroller modification is a minor change, o Errata analysis shall be performed in order to determine the level and nature of faults detected. This analysis should reveal the level of maturity of V&V performed by the SoC suppliers, o Visibility of the SoC evolution road map to assess the continuous wide diffusion and to prevent risks of obsolescence • Applicant experience in the use of the SoC microcontroller should be demonstrated by: o Past experience on similar core or similar SoC family, o Current development: number of hours for verification, endurance, stress testing	Step 4: The quality of service experience data may not be adequate to meet certification objectives alone. Service experience may supplement other design assurance means. The SoC microcontroller, which has been in the market for at least 2 years, has been sold to millions and is more robust because it has been extensively tested by being used in various application fields. This is not the case for newly released SoC microcontrollers because they have only been tested by the manufacturer with sufficient test scenario for first commercial release. Assuring aircraft safety requires the number of errors to be minimal. It means that the number of errata is decreasing after each new silicon revision. In order to meet the time to market objective, a first version of the silicon (e.g. beta version) is launched before complete testing, with the objective to use the end user application as debugging.

2 – SoC microcontroller selection process			
Objectives	Reference	Activities	Rational – Output of public/private data
SoC microcontroller configuration changes	ED80/DO254 §7.2.3 Problem reporting, tracking and corrective action,	The applicant should assure quality and performance of all components used throughout the production cycle. • SoC microcontroller variation parameters should be controlled by appropriate means (i.e.: incoming testing, screening, statistical process control...) • A SoC microcontroller design and manufacturing changes process should be implemented between the applicant and the SoC manufacturer throughout the life cycle of the system: o The process of tracking and monitoring SoC microcontroller design, manufacturing process, data change shall be documented and implemented even if there is no change of the P/N. o Following analysis of the SoC changes, the applicant should determine potential effects on the operation of the system. • Effectiveness of Problem reporting activity: o Problem reporting system of the SoC microcontroller supplier shall be assessed, including their public use.	Step 4 Frequent design and manufacturing changes are made to improve yield, to reduce cost and to enhance performance. Although these changes are documented through a PCN/PCI, the applicant should continuously track any changes (design, data...) and assess their effects on the application.

3 – Safety Analysis			
Objectives	Reference	Activities	Rational – Output of public/private data
Functional failure path analysis	ED80/DO254 Appendix B §2	The applicant should perform a functional failure path analysis within the SoC microcontroller. The FFPA may consist of: • Identifying the internal SoC hardware function (also called FFP), • Identifying interrelationship between SoC hardware function, • Identifying the specific portion of the design which implements the internal SoC hardware function, • Performing a failure mode and effects analysis considering: o Each internal SoC hardware functions potential loss or abnormal behavior (due to potential design error or upset), o Potential inadvertent activation of unused functions, o Internal safety mechanism (ECC, parity). • Assigning a design assurance level that specifically correlates to the failure conditions that the internal hardware function is capable of causing. • Checking for potential common modes between the SoC hardware function that may compromise their assigned Design Assurance levels Selecting Design Assurance strategy for FFP level A, B (refer to [4]).	Step 2 To use a SoC microcontroller within an equipment with a Design Assurance Level A or B without any precaution in the design assurance strategy or in the architecture is not conceivable. An FFPA may be used to justify assigning a lower Design Assurance level for the entire SoC microcontroller or some FFPs within it, when an assessment of the consequences of its loss or abnormal behavior shows that the higher Design Assurance Level is not degraded. Decomposition is performed using conventional top-down safety assessment techniques and may be complemented by bottom-up analysis methods for each successive level of decomposition.

4 – Design assurance			
Objectives	Reference	Activities	Rational – Output of public/private data
Safety-specific analysis	ED80/DO254 Appendix B §3.3.2	The applicant should perform a safety-specific analysis for identified level A and B FFPs. The safety-specific method may consist in • Identifying safety-sensitive functions of the SoC which are to be addressed by safety-specific methods. • Determining safety-sensitive attributes that need to be addressed by safety-specific conditions. • Identifying their potential anomalous behavior which can cause a concern for safety. • Identifying relevant observable detection means provided by the SoC microcontroller (Debug, performance monitoring), • Identifying input verification condition and the associated expected output behavior to detect potential anomalous behavior. • Performing verification addressing the input verification condition and output space behavior to be verified, • Bounding the input set and condition on the implementation so that it is not possible that the SoC receives inputs outside the tested condition.	Step 3: The purpose of these activities is to expose design errors that could adversely affect the hardware output and therefore have a critical safety impact. The design error can be detected only when specific input stimuli are exposed. Safety specific attributes could be logic decision, computation, timing, state transition and real time attributes.

4 – Design assurance			
Objectives	Reference	Activities	Rational – Output of public/private data
ED12B/DO178B	ED12B/DO178B	The applicant could base the design assurance credit on specific functions of software design activities based on ED12B/DO178B • Identification of processor core functions for which the credit can be claimed on ED12B/DO178B activities, • Justification that design errors of these processor core functions are precluded by ED12B/DO178B activities, • Additional means of design assurance should be given for part of or the whole processor for which ED12B/DO178B cannot be claimed.	Step 1 Software solution alone may not address the problem of the hardware functions implemented in the processor core. Credit on ED12B/DO178B could be claimed only on the processing part of the processor (ALU/FPU, Load/store unit, Instruction cache). Credit of ED12B/DO178B for MMU, dynamic branch prediction, DPM functionalities could not be possible.
Simplicity	ED80/DO254 §1.6	The applicant could base the design assurance credit on simplicity of function within the FFP: • Identify simple functions, • Perform comprehensive testing on simple FFP.	

5 – Architectural mitigation			
Objectives	Reference	Activities	Rational – Output of public/private data
Revise the system architecture	ED80/DO254 Appendix B §3.1 ARP4754 ARP4761	The applicant should modify and revise the system architecture with an appropriate mitigation technique, if the SoC microcontroller implementation can not provide acceptable Design Assurance for FFP level A, B. The applicant shall revise the architecture by implementing mitigation mechanisms until an acceptable strategy of Design Assurance has been determined. • Identification of FFPs that have to be protected by architectural means, • Capture specific safety-related requirements such as containment boundary definitions, partitioning strategies at the appropriate levels (HW, SW, system), • Update the PSSA, CMA with the new mitigation means. The mitigated hardware FFPs could not lead to a failure condition CAT or HAZ.	Failure detection due to design error may be accomplished through dedicated circuitry, software code, system architecture etc.

6 – SoC supplier assistance			
Objectives	Reference	Activities	Rational – Output of public/private data
Cooperative assistance	ED80/DO254 § 4.2 Item 9	This design information may be available from the SoC microcontroller supplier. As this design information is usually submitted to confidentiality, the applicant shall at least demonstrate evidence of collaboration and shall be responsible for the use of this data. The applicant should at least: • Describe the cooperative approach which is implemented. Define the type of data which will be shared. • Describe how this data will be used in SoC activities • Provide evidence of data accessed in case of cooperative approach: o Meeting reports including the type of information provided, o Impact on design activities (HW, SW, System). • Provide a cooperative approach summary	Some SoC microcontroller manufacturers might not agree to provide any sensitive information about their products. The risks of a SoC microcontroller manufacturer refusal to provide information should also be considered and taken into account from the planning phase and addressed in the PHAC. Private data assessment: The public data analysis highlighted the lack of information to get sufficient confidence for any use within safety critical application. Adequate detailed design information at the elemental level will be required to complete at least the FFPA in order to determine an appropriate design assurance strategy.

Table 7 - Activities for SoC microcontroller use

It will be the responsibility of the applicant to explain in which data/document, related to Appendix A of the ED80/DO254, the above recommendations will be captured to demonstrate the evidence of completion of these activities. However, the hardware/software interface document should be a key document to capture the SoC configuration.

6.2 SOPC RECOMMENDATIONS

This survey shows that it may be difficult to meet the ED80/DO254 objectives when designing a SoPC. Indeed, as long as the SoPC involves IPs as black boxes, current certification recommendations can not be met. Moreover, even in the case of a full custom SoPC, it may be necessary to clarify or reinforce some recommendations of the certification memo [R4] to ensure a safe SoPC design.

Then, the idea here is to refine some current certification recommendations and to propose additional ones in such a way that the SoPC certification recommendations:

- Are based on the ED80/DO254 and the certification memo,
- Take into account the potential weaknesses and issues highlighted in section 4,
- Take into account the possibility to use third party IPs with or without ED80/DO254 life cycle data,
- Take into account the possibility to use software, firmware or hardware IPs,
- Take into account the potential evolutions of technologies and tools.

The principle of these recommendations is to introduce a hierarchical level in the SoPC architecture definition in order to have a better visibility inside the SoPC and master its different constituents. Thus, each IP which constitutes the SoPC should be considered as a stand alone hardware item and should be designed according to the recommendations of the certification memo [R4].

The following figure illustrates the proposed approach. It focuses mainly on the hardware design but also makes the link with the software design process which should be able to run in the SoPC environment. Before applying the following flow chart, it is assumed that the expected functionalities of the SoPC have been defined and a hardware / software allocation has been performed.

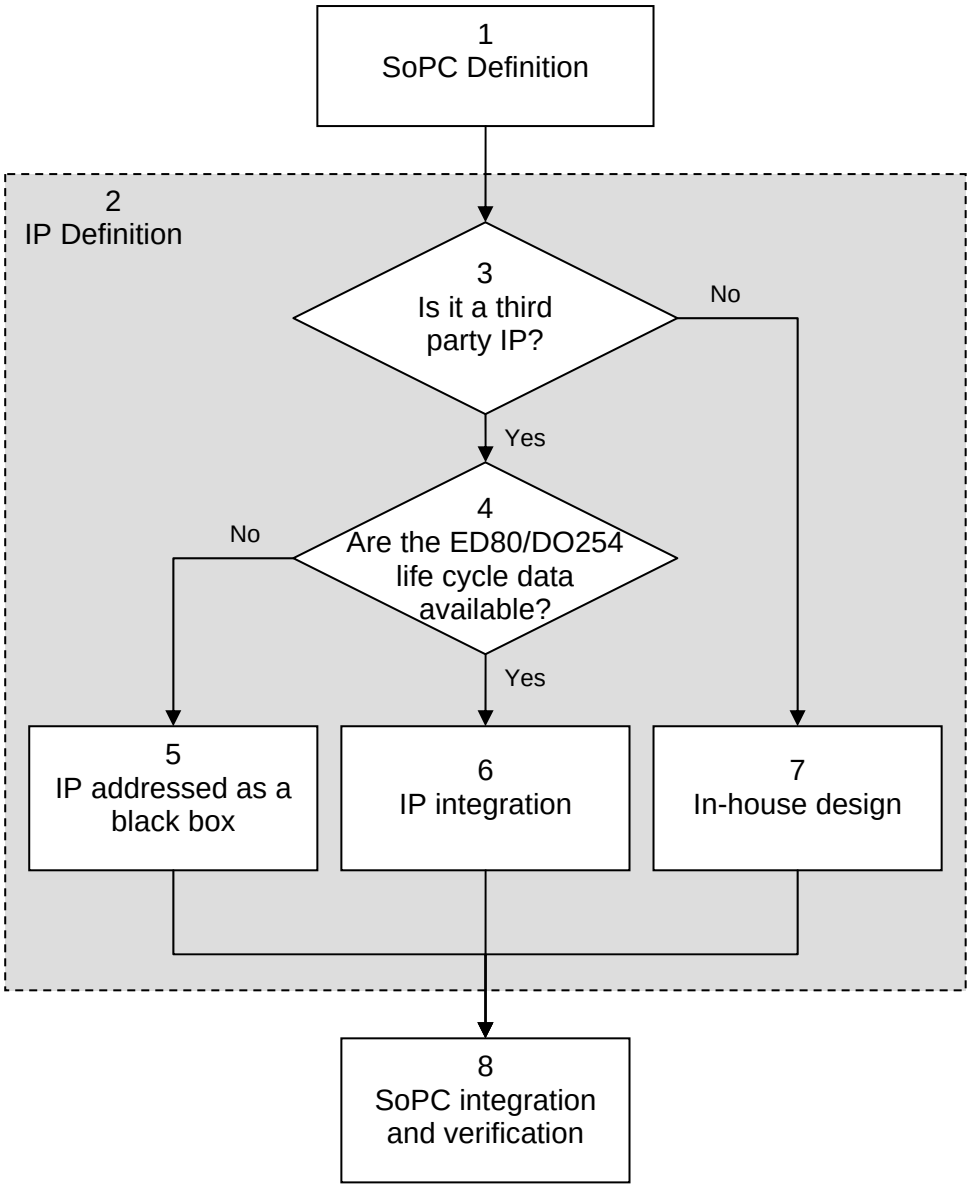


Figure 8 – Proposed approach for SoPC design

Note: The approach depicted above is applicable for Design Assurance Level A, B, C and D. The activities and data to produce for each IP can be adapted according to the actual IP Design Assurance Level.

1 - SoPC definition

SoPC definition includes the requirements capture and conceptual design activities at SoPC level (device level), and associated supporting processes as defined in the ED80/DO254 and certification memo. The entry data of this phase are the functionalities allocated to the hardware.

It is essential that the complexity and the modularity of the SoPC are taken into account from the requirements capture phase. Indeed, the modular approach that consists in building a SoPC by interconnecting IPs (previously designed or not, with or without ED80/DO254 life cycle data) on an internal system bus requires that the characteristics of the SoPC internal architecture are addressed as requirements. Thus, the interoperability of the various IPs will be ensured since verified on the basis of those requirements.

The following figure illustrates what should be defined as a minimum during this phase.

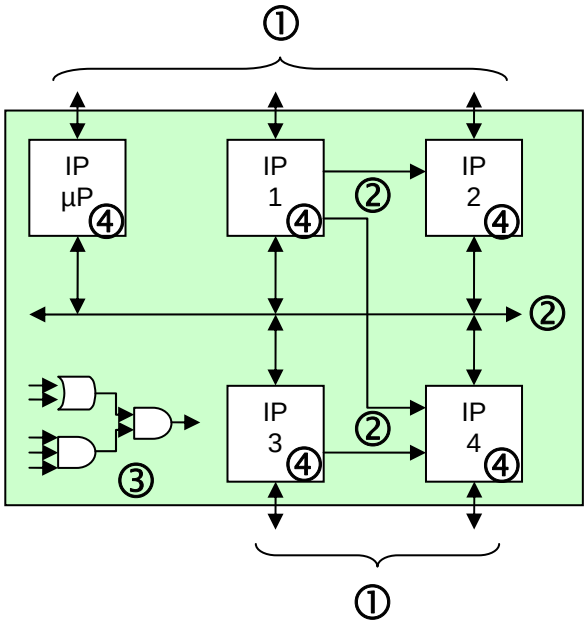


Figure 9 – SoPC architecture definition during phase 1

Item	Item label	Characteristics to define
①	SoPC pinning External pinning of the device	Signals Logical behavior Timing Electrical level
②	Internal signals Internal signals that participate to the communication between the IPs. They can be part of internal communication busses or discrete signals. Note: Internal communication bus can be addressed as a specific IP that connects several IPs.	Signals Logical behavior Timing

Item	Item label	Characteristics to define
③	Glue logic Any function that can not be attached to a specific IP and which should be implemented at SoPC level. It includes but is not limited to: • Clock and reset trees, • Active logic level adaptation, • Signals multiplexing and copying for testability purpose (Cf. §6.2.1 Verification consideration).	Signals Function to realize Timing
④	IPs Definition of the IP seen as a black box	Pinning of the IP (Signals, logical behavior, timing) Function to realize

Table 8 - Constituents to define during the SOPC definition

The outputs of this phase are:

- SoPC specification at device level (functionality, pinning and architecture definition) and associated traceability with the upper level requirements. It includes derived requirements that result from the conceptual design,
- Conceptual design data that include the above details,
- Validation data,
- Feedback of the derived requirements to the safety analysis.

At this stage, the SoPC design choices have an impact on the software (basically memory and register mapping, but also rules to access a specific hardware function). The expected software behavior should be defined and captured in a Hardware Software Interface Document (HSID), which should be an entry document for the software specification.

2 – IP definition

IP definition defines the activities to perform for each IP, according to the design assurance strategy which depends on the IP nature.

Whatever the IP nature, their definition or their use may have a software impact that should complete the HSID.

Note: At this stage, the Design Assurance Level of a specific IP can be lowered provided the commonly agreed segregation rules are applied and the applicant demonstrates that the IP can not impact a higher Design Assurance Level IP.

3 - Is it a third party IP?

Is the IP provided by a third party or is it an in-house IP?

Firmware IPs provided as part of PLDs vendor's tools, and hardware IPs embedded in the PLD silicon are third party IPs.

A software IP, provided by a third party, which is intended to be reverse engineered to produce a complete ED80/DO254 design life cycle data should be addressed as an in-house IP.

4 - Is the ED80/DO254 life cycle data available?

Is the IP delivered with a design life cycle data package in line with the recommendations of the ED80/DO254? The life cycle data should include at least:

- The specification,
- The validation data (if relevant),
- The conceptual design data,
- The detailed design data (source code),
- The demonstration data of the consistency between source code, design data and specification,
- The traceability data between requirements, design data and source code,
- RTL level verification test benches and results, and the associated traceability with the requirements,
- Open Problem Reports list,
- The HDL design standard applied to produce the source code (if applicable).

Note: Post synthesis and post Place and Route verification results are not required here. Indeed, an applicant could not take credit of these results since the final IP implementation within the SoPC will not be necessarily the same than the implementation for which it was initially designed.

5 - IP addressed as a black box

In this case, the IP is addressed as a black box and the IP should be assessed through the strategy proposed for SoC microcontrollers (cf. §6).

As defined in §6, the applicant should:

- Define the IP usage domain,
- Assess the IP against the selection process,
- For levels A and B, perform a safety analysis including assessment of the SoPC FFPs which involves the IP.

Those activities may require to be supported by the IP supplier and may require implementing an architectural mitigation, if the design assurance strategy is not deemed relevant for the IP.

Activities and data to produce depend on the chosen design assurance strategy (cf. §6 for more details). As a minimum, such an IP should be the object of implementation requirements at SoPC level.

6 - IP integration

The integration of a third party IP delivered with ED80/DO254 life cycle data should include at least the following activities:

- The IP specifications are part of the whole SoPC specification architecture (establishment of the traceability with SoPC level requirements),
- Validation of IP requirements against SoPC level requirements,
- Assessment of IP derived requirements against safety objectives (IP requirements that can not be linked to an upper requirement should be addressed as derived requirements),

- Analysis of the IP Open Problem report against the safety objectives and the expected IP behavior,
- Configuration Management of the IP data package.

Note: It is the responsibility of the applicant to assess the adequacy of the life cycle data package with the ED80/DO254 objectives. In the same way, according to the certification basis of the SoPC, the IP data package may need to be completed with additional data or activities.

Note: A third party IP may involve unnecessary functions for the final SoPC application. In this case, the applicant should demonstrate that the unused function is managed through specific requirements at SoPC or IP level, and that it has no impact on the used functions of the IP. In case of deactivation or isolation of the unused function, the applicant should either demonstrate that the unused function can not be inadvertently activated, or that it has no impact on the used functions in case of inadvertent activation.

The outputs of this phase are:

- IP requirements traceability with SoPC requirements,
- Validation data,
- Open Problem report analysis

7 – In-house design

The IP should be designed according to the ED80/DO254 and certification memo recommendations, taking into account the design assurance level allocated to the IP.

The outputs of this phase are (if relevant according to the design assurance level):

- IP specification updated with derived requirements issued from the IP conceptual design activity,
- Validation data,
- Feedback of derived requirements to safety analysis,
- IP conceptual design data,
- IP detailed design data (source code),
- Demonstration of coherence between requirements, design data and source code,
- Demonstration of conformance to HDL design standard (if applicable),
- Traceability data between requirements, design data and source code,
- RTL simulation and code coverage measurement (if applicable).

Note: The RTL simulation and the eventual code coverage measurement can be performed during simulation at IP level and/or at SoPC level (cf. point 8).

The physical verification (post Place and Route tests) should be performed at SoPC level (cf. point 8).

8 – SoPC integration

SoPC integration includes the following activities at SoPC level:

- The detailed design production (source code at SoPC level: IP instantiation and glue logic code production),
- The implementation (synthesis and Place and Route),
- The verification.

The purpose of the verification is to perform the verification activities at SoPC level and to complete the verification activities at IP level.

At SoPC level, the verification activities consist in:

- Verification of SoPC requirements by simulation at RTL level on the complete SoPC HDL code. It includes the code coverage measurement for level A and B of the top level HDL module (it is not necessary to measure the code coverage of the IPs already simulated at RTL level, and for which the code coverage has already been measured),
- The physical verification of the SoPC requirements on the final target device. For requirements that can not be verified on the final target, an alternative mean of verification should be proposed,
- The analysis of the synthesis and Place and Route reports,
- The analysis of the Static Timing Analysis results.

At IP level, the verification activities consist in:

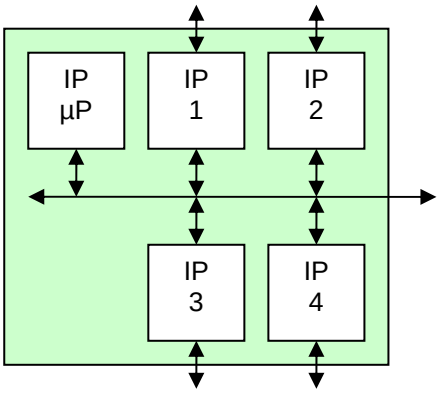
- For the IPs that have not yet been verified by simulation at RTL level, verification of IP requirements by simulation at RTL level on the complete SoPC HDL code (including code coverage measurement of the IP source code, for Design Assurance Level A and B),
- Physical verification of all the IP requirements on the final target device. For requirements that can not be verified on the final target, an alternative mean of verification should be proposed.

The outputs of this phase are:

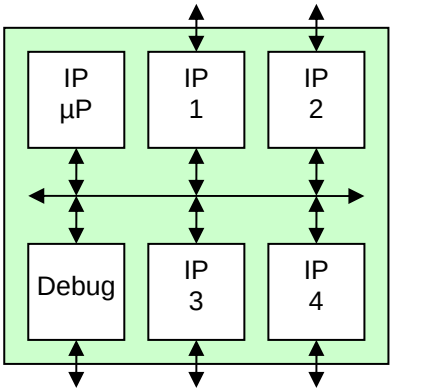
- Detailed design data at SoPC level,
- Demonstration of coherence between requirements, design data and source code,
- Demonstration of conformance of the SoPC source code to HDL design standard (if applicable),
- Traceability data between requirements, design data and source code,
- RTL simulation and code coverage measurement,
- Verification procedures and results at SoPC and IP levels, and traceability with requirements.

6.2.1 Verification consideration

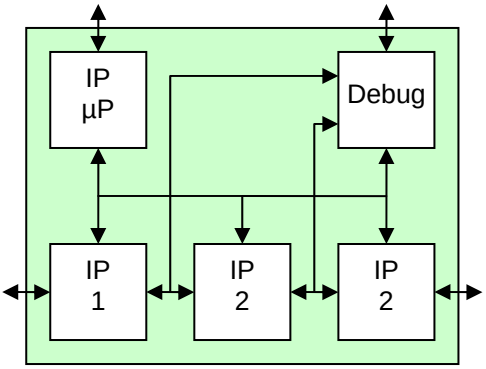
It is essential that the testability of each IP on the final target device is taken into account from the SoPC requirements capture phase. Indeed, the verification of the IP may require an intrusive access on the internal signal of the SoPC. Thus, it may be necessary to implement in the SoPC one or several modules dedicated to the IP verification activities. It is the case when the internal architecture of the SoPC does not allow to individually verify its different IPs at physical level. The following figures illustrate various possibilities.



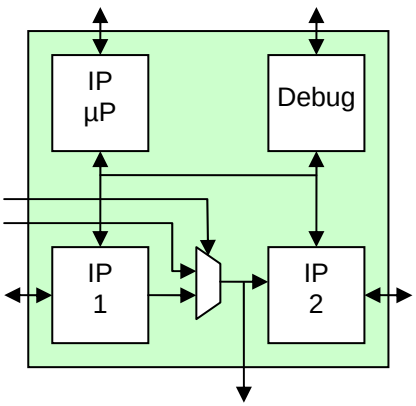
Physical testability of all the IPs is ensured through direct external access to the internal bus and IPs I/O that allow to stimulate and spy all the internal signals.



A debug module is added in order to ensure an external access to the internal bus and thus, to stimulate and to spy the various IP.



A debug module is added in order to spy the internal bus and the internal signals between the IPs. The IP µP involves its own debug function that allows the user to take hand on the internal bus and stimulates the IPs.



A debug module is added in order to spy the internal bus. Other internal signals are outputs for monitoring and various IPs inputs are multiplexed in order to allow an external stimulation.

Figure 10 – Testability architecture

In the context of additional modules and/or logic dedicated to verification activities, consideration on embedded parts without operational functions should be applied. It means that an applicant should either demonstrate the same rigor of development process for additional module/logic than for the rest of the SoPC, either demonstrate that, in operational mode, the additional parts are isolated from the rest of the SoPC, or deactivated, in such a way that they do not jeopardize the operational functions.

The verification activities dedicated to each IP should be adapted to the Design Assurance Level allocated to each IP. It means that for level A and B IPs, the certification memo recommendations related to ED80/DO254 Appendix B should be applied. In the same way, certification memo recommendations on verification independence should be applied.

6.2.2 Tools consideration

The certification memo recommendations related to the tools should be applied. It means that if a non qualified tool is used to automatically generate the SoPC source code or to configure an IP, the resulting source code should be manually reviewed. If impossible, the use of such a tool should be avoided.

It includes, but is not limited to:

- The “SoPC builder” tools proposed by the various SoPC providers that allow to design a complete SoPC, or a part of the SoPC, from a graphical interface, and automatically produce the source code of the design,
- The wizard tools that allow to configure a specific IP and that automatically generate the generic parameters or customize the source code.

6.2.3 Configuration management consideration

The recommendations of certification memo related to generic ASIC/PLD should be applied to the previously designed IPs.

Care should be taken on IP libraries included in PLD vendor’s tools. If the version of such IPs are implicitly managed by the tool (the IP version depends on the tool version), the applicant should propose an alternative mean, independent to the vendor's tool, to manage the IP configuration.

6.2.4 SEU Management Consideration

A Single Event Upset analysis should be performed as described in the certification memo. For this analysis, the applicant should take into account the various possible design implementations and the related options of the design tools.

Indeed, an IP may or may not use SEU sensitive resources of the PLD according to the synthesis tool options. In this context, an IP may be reputed to be not SEU sensitive according to synthesis options or according to the PLD target technology, and may be SEU sensitive for other synthesis options or PLD technologies.

7. CONCLUSION

This survey has shown that the current design assurance standards used as acceptable means of compliance to comply with CS or FAR (ED80/DO254 §11.2, §11.3 and ED12B/DO178B for microprocessor) cannot give a sufficient level of confidence to implement SoC microcontrollers or SoPCs based on IPs within safety critical systems.

This situation should not get any better in the future with the ever growing complexity and integration of embedded electronics.

A new approach has been proposed in this survey to counteract these safety and certification issues.

The success of this approach relies on:

- The applicant's rigor to implement the SoC/IP within its application.
- The cooperation between the applicant and SoC/IP providers in order to share confidential information.
- This item could be a concern and should be assessed by the applicants during the planning phase. The PHAC should explain and demonstrate how the applicant expects to collaborate with the SoC/IP provider and what kind of evidence he/she will be able to present to the certification authorities.
- The adaptation of the current PLD design process to take into account design techniques based on integration of pre-qualified hardware blocks.

The survey was also the opportunity to work with SoC/IP providers and certification authorities. It highlighted the fact that both parties should be more aware of each other's work in order to facilitate the use of latest microelectronic technologies while ensuring appropriate safety levels.