GAMMA
Generic Architecture for mass memory Access
(Contract ESTEC 17437/03/NL/AG)

GAMMA study executive summary

<table>
<thead>
<tr>
<th>Written by</th>
<th>Name</th>
<th>Company</th>
<th>Signature</th>
<th>Internal reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Christophe Honvault</td>
<td>EADS Astrium SAS</td>
<td>AOE7.RA.HON.10424.05</td>
<td></td>
</tr>
</tbody>
</table>

GAMMA (Generic Architecture for mass memory Access) is an ESA project (Contract ESTEC 17437/03/NL/AG) conducted by a consortium led by EADS Astrium SAS with EADS Astrium GmbH and EADS Astrium Ltd. For more information please contact:

Michael Schön
ESTEC. Keplerlaan 1. PO Box 299
2200 AG Noordwijk ZH – The Netherlands
Tel: +31 (0) 71 565 6823. Fax: +31 (0) 71 565 5420
e-mail: michael.schoen@esa.int

EADS Astrium SAS
Christophe Honvault
EADS Astrium SAS
31, avenue des cosmonautes
F-31 402 Toulouse Cedex 4, France
Tel: +33 5 62 19 70 90. Fax: +33 5 62 19 78 97
e-mail: christophe.honvault@astrrium.eads.net

EADS Astrium GmbH
Mladen Kerpe
EADS Astrium GmbH
An der B 31
88090 Immenstaad
Tel: +49 75 45 8-2736. Fax: +49 75 45 8-3332
e-mail: mladen.kerpe@astrrium.eads.net

EADS Astrium Ltd
Les Griffiths
EADS Astrium Ltd
Anchorage Road
Portsmouth – Hampshire PO3 5PU
Tel: +44/23-92705869. Fax: +44/23-92708290
e-mail: les.griffiths@astrrium.eads.net
ABSTRACT:

GAMMA is an ESA project that aims to define the Data Management Software architecture suitable to support all the payload and platform requirements in term of data storage access for any type of mission and develop a prototype to validate the architecture and measure its performances.

The proposed architecture is layered, distributed and independent from the mass memory technology. It supports a transparent memory storage partitioning leading to increase its reliability, security and performance. The layered architecture eases the integration of COTS components while ensuring their isolation in order to avoid fault propagation. GAMMA can also take advantage of the development of hardwired functions to support the storage and ensure the consistency of the data in a distributed system.

The instantiation of the architecture is made on a representative and versatile Hardware platform. This platform is based on five commercial boards equipped with a six million gates FPGA and 512 Mbytes of memory. Each board can be configured as a memory user (processor, instrument or telemetry encoder) or as a memory module of up to 4 Gbits. The FPGA is programmed with a LEON 2 processor, two SpaceWire interfaces and the CCSDS Time Manager. The boards configured as memory modules also include a specific set of hardwired functions that manage the SpaceWire communication. Support configuration commands, provide data access and protection services. These functions are developed using a high-level language: HandelC and tested during a co-simulation step. After successful simulation, the functions are placed on the FPGA.

The current implementation of GAMMA Software includes one COTS file management library and two specific data management libraries. The entire storage area provided by one or several memory modules can be decomposed into logical partitions. Each partition can be managed by a distinct file management library. The GAMMA Software provides all the Application Programming Interfaces required to manage partitions, directories and files while ensuring the consistency of the stored data in a distributed system.

In addition to traditional integration and validation tests, several applications have been developed to cover the entire range of use for the data storage system. These applications are executed in the frame of scenarios that are representative of present and future earth observation, deep-space and telecommunication missions.

At the end of the project, the results obtained showed the correct behaviour of the GAMMA architecture in a distributed environment and very interesting performance with respect to the prototype configuration.

Deliverables:

- Requirements Specification Document
- Reference Scenarios Specification
- Prototype Implementation Proposition
- Detailed Performance Analysis of Reference Scenarios
- Prototype implementation associated documentation
- Software and demonstration platform user manual
- Validation report and associated documentation
- Final presentation slides
- User Services and Primitives Specification
- System Analysis and Architectural Design
- Architectural Detailed Design Document
- Executive summary (this document)

The work described in this report was done under ESA contract. Responsibility for the contents resides in the author or organisation that prepared it.

Name of author: Christophe Honvault (EADS Astrium SAS)
Table of Contents

1  Introduction .......................................................................................................................... 5
   1.1  Scope .................................................................................................................................. 5
       1.1.1  Scope of the Project .................................................................................................. 5
       1.1.2  Scope of the Document .......................................................................................... 5
   1.2  Related Documentation .................................................................................................. 5
   1.3  Definition of Terms and Acronyms .............................................................................. 6
       1.3.1  Definition of terms ................................................................................................. 6
       1.3.2  Acronyms and abbreviations .................................................................................. 7
2  Introduction .......................................................................................................................... 8
   2.1  Overview .......................................................................................................................... 8
   2.2  Context ................................................................................................................................ 9
   2.3  Organization ..................................................................................................................... 10
3  Phase 1 ................................................................................................................................... 12
   3.1  Requirements and constraints ...................................................................................... 12
   3.2  User services and primitives ......................................................................................... 16
   3.3  Reference scenarios ......................................................................................................... 17
       3.3.1  Earth Observation mission ......................................................................................... 17
       3.3.2  Deep Space mission .................................................................................................. 19
       3.3.3  Telecommunication mission ...................................................................................... 21
   3.4  System Analysis and Architecture .................................................................................. 23
4  Phase 2 ................................................................................................................................... 28
   4.1  System and Software Architecture Detailed Design ..................................................... 28
   4.2  HW/SW Detailed Design and related performance evaluation ....................................... 29
   4.3  Prototype Implementation .............................................................................................. 30
       4.3.1  GAMMA prototype elements ................................................................................. 30
       4.3.1.1  Memory users and modules .................................................................................. 30
       4.3.1.2  Network ................................................................................................................. 30
       4.3.2  GAMMA prototype configuration ............................................................................. 32
       4.3.3  GAMMA prototype implementation ......................................................................... 33
   4.4  Prototype Verification ..................................................................................................... 35
       4.4.1  Functional results ....................................................................................................... 35
       4.4.2  Performance results ................................................................................................... 35
   4.5  Assessment, Evaluation and Future Directions ............................................................... 39
       4.5.1  Overall study assessment .......................................................................................... 39
       4.5.2  Functional assessment ............................................................................................... 41
       4.5.3  Performance assessment ............................................................................................ 41
       4.5.4  GAMMA future directions ......................................................................................... 42
           4.5.4.1  Extension and optimization ................................................................................ 42
           4.5.4.2  FDIR Assessment .............................................................................................. 42
           4.5.4.3  Integration in validation facilities ......................................................................... 43
           4.5.4.4  Technology transfer ............................................................................................ 43
5  Conclusion ............................................................................................................................ 44
List of Figures

Figure 1: GAMMA Phase 1 .................................................................................................................. 11
Figure 2: GAMMA Phase 2 .................................................................................................................. 11
Figure 3: Main missions analyzed......................................................................................................... 13
Figure 4: Earth Observation reference architecture........................................................................... 18
Figure 5: Earth Observation reference scenario ................................................................................ 18
Figure 6: Deep Space reference architecture..................................................................................... 20
Figure 7: Deep Space reference scenario ............................................................................................ 20
Figure 8: Telecommunication reference architecture......................................................................... 22
Figure 9: Telecommunication reference scenario .............................................................................. 22
Figure 10: Mass memory environment ............................................................................................... 23
Figure 11: Mass memory physical organization ................................................................................. 23
Figure 12: Physical network ................................................................................................................. 24
Figure 13: Mass memory system network ........................................................................................... 24
Figure 14: Partition layout ..................................................................................................................... 25
Figure 15: The wrapper ......................................................................................................................... 25
Figure 16: The Generic Architecture for Mass Memory Access.......................................................... 26
Figure 17: GAMMA architecture adaptability ..................................................................................... 26
Figure 18: General view of the layered architecture .......................................................................... 27
Figure 19: areaLock activities ............................................................................................................... 28
Figure 20: The event management ..................................................................................................... 29
Figure 21: The FPGA board used to implement memory users and memory modules ..................... 30
Figure 22: The SpaceWire router ........................................................................................................ 31
Figure 23: The 2 RS232/2 SpaceWire interface board ........................................................................ 31
Figure 24: GAMMA prototype configuration used for validation ...................................................... 32
Figure 25: GAMMA prototype configuration assembled ..................................................................... 32
Figure 4-26: GAMMA implementations ............................................................................................ 33
Figure 4-27: The memory module configuration ............................................................................... 34
Figure 28: GAMMA primitive building block architecture ................................................................. 34
Figure 29: Target boards memory mapping ......................................................................................... 35
Figure 30: SpaceWire communication data rates ............................................................................... 35
Figure 31: Memory module data rates ................................................................................................ 36
Figure 32: ERTFS and PRBFS performance (one or two users accessing a memory module) .......... 37
Figure 33: RAWFS and PRBFS performance (one or two users accessing a memory module) ....... 37
Figure 34: Performance summary ...................................................................................................... 38
Figure 35: Repartition of time .............................................................................................................. 38
Figure 36: The co-design process ........................................................................................................ 39
Figure 37: Hardware/Software decomposition .................................................................................... 39
Figure 38: Co-design planning ............................................................................................................ 40
Figure 39: GAMMA configurations ................................................................................................... 41
1 Introduction

1.1 Scope

1.1.1 Scope of the Project

GAMMA (Generic Architecture for mass memory Access) is an ESA project (contract ESTEC 17437/03/NL/AG). The objectives of the project are to define the Data Management Software architecture suitable to support all the payload and platform requirements in term of data storage access for any type of mission, make the architecture independent from the mass memory technology and compatible with the use of COTS software, develop a prototype to validate the design.

The project is split in two phases. The first phase is dedicated respectively to the Requirements Capture and Consolidation (radar and optical observation and science missions as well as platforms), the specification of User Services and Associated Service Primitives, the System Analysis and Architectural Design and the proposition of a prototype implementation. The second phase consists in the System Detailed Design, Prototype implementation, verification and delivery at ESTEC and a global assessment and evaluation of the results of the study with respect to the identified needs and expected performances, and the identification of future directions.

1.1.2 Scope of the Document

This document is the GAMMA study executive summary (EXS). It aims to summarize the result of the GAMMA study. It reminds the main objectives and requirements of the study. Then, it presents the implementation of the architecture that have been performed and an assessment with respect to the requirements. The results of the different tests on the GAMMA demonstration platform are summarized and a conclusion of the study is presented.

After this introduction, this document is split in four chapters. The chapter 2 presents a global view of the project and presents the organization of the project and the content of its two phases. The chapter 3 presents the work performed in the frame of the first phase that includes the requirements and architecture tasks. The chapter 4 presents the work performed in the frame of the second phase that includes the detailed design, the implementation and the validation activities. A conclusion of the study is given in the chapter 5.

1.2 Related Documentation


[GAMMA Prop] Data Management Software for mass memory Based Payload Processors, Astrium SAS/GmbH/Ltd, Proposal for ESA ITT AO/1-4220/03/NL/AG, March 2003, SE13.PC.VC.8068.03.


[GAMMA PIP] GAMMA Prototype Implementation Proposition, EADS Astrium SAS, Issue 1.0, June 2004, AOE7.TCN.90112.ASTR.
Mass memory: High capacity memory equipment supporting the storage of all the spacecraft generated data. Generally used to store the data generated by payloads instruments, the use of mass memory now extends to the platform requirements (storage of Interpreted Procedures, Time-tagged Telecommands, etc.).

SpaceWire: Network for space applications composed of nodes and routers interconnected through bidirectional high-speed digital serial links\(^1\). The main purpose of the SpaceWire Standard is to ensure

\(^1\) http://www.estec.esa.nl/tech/spacewire/
the compatibility between different equipment, in order to facilitate integration, testing and reduce
time-to-market and cost and its re-use in different missions\(^2\).

**SGDR** (SafeGuard Data Recorder): ESA study on a safe mass memory. The function of the Safeguard
Data Recorder (SGDR) in the system is to provide a data storage service implemented in a way such
as it tolerates specific system level events or failures without loosing data.

The term SGDR includes all the hardware and software modules implementing the function, and
particularly the software layer running on the user side (e.g. as part of a satellite central computer
software).

### 1.3.2 Acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>American National Standard Institute</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application-Specific Integrated Circuit</td>
</tr>
<tr>
<td>CCSDS</td>
<td>Consultative Committee for Space Data Systems</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial (or Component) Off-The-Shelf</td>
</tr>
<tr>
<td>DHS</td>
<td>Data Handling System</td>
</tr>
<tr>
<td>DMS</td>
<td>Data Management System</td>
</tr>
<tr>
<td>ECSS</td>
<td>European Cooperation for Space Standardisation</td>
</tr>
<tr>
<td>FDIR</td>
<td>Fault Detection, Isolation and Recovery</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>GAMMA</td>
<td>Generic Architecture for mass memory Access</td>
</tr>
<tr>
<td>HKTM</td>
<td>Housekeeping Telemetry</td>
</tr>
<tr>
<td>IP</td>
<td>Interpreted Procedure (equivalent to OBCP)</td>
</tr>
<tr>
<td></td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>kbps</td>
<td>Kilobit per second</td>
</tr>
<tr>
<td>OBCP</td>
<td>On-Board Control Procedure (equivalent to IP)</td>
</tr>
<tr>
<td>POSIX</td>
<td>Portable Operating System Interface for Unix</td>
</tr>
<tr>
<td>PUS</td>
<td>Packet Utilization Standard</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real-Time Operating System</td>
</tr>
<tr>
<td>SOW</td>
<td>Statement of Work</td>
</tr>
<tr>
<td>SSMM</td>
<td>Solid State Mass Memory</td>
</tr>
<tr>
<td>TC</td>
<td>Telecommand</td>
</tr>
<tr>
<td>TM</td>
<td>Telemetry</td>
</tr>
<tr>
<td>TN</td>
<td>Technical Note</td>
</tr>
<tr>
<td>WCET</td>
<td>Worst Case Execution Time</td>
</tr>
</tbody>
</table>

\(^2\) [http://www.estec.esa.nl/tech/spacewire/standards/](http://www.estec.esa.nl/tech/spacewire/standards/)
2 Introduction

2.1 Overview

GAMMA is an ESA project that aims to define the Data Management Software architecture suitable to support all the payload and platform requirements in term of data storage access for any type of mission and develop a prototype to validate the architecture and measure its performances.

The proposed architecture is layered, distributed and independent from the mass memory technology. It supports a transparent memory storage partitioning leading to increase its reliability, security and performance. The layered architecture eases the integration of COTS components while ensuring their isolation in order to avoid fault propagation. GAMMA can also take advantage of the development of hardwired functions to support the storage and ensure the consistency of the data in a distributed system.

The instantiation of the architecture is made on a representative and versatile Hardware platform. This platform is based on five commercial boards equipped with a six million gates FPGA and 512 Mbytes of memory. Each board can be configured as a memory user (processor, instrument or telemetry encoder) or as a memory module of up to 4 Gbits. The FPGA is programmed with a LEON 2 processor, two SpaceWire interfaces and the CCSDS Time Manager. The boards configured as memory modules also include a specific set of hardwired functions that manage the SpaceWire communication, support configuration commands, provide data access and protection services. These functions are developed using a high-level language: HandelC and tested during a co-simulation step. After successful simulation, the functions are placed on the FPGA.

The current implementation of GAMMA Software includes one COTS file management library and two specific data management libraries. The entire storage area provided by one or several memory modules can be decomposed into logical partitions. Each partition can be managed by a distinct file management library. The GAMMA Software provides all the Application Programming Interfaces required to manage partitions, directories and files while ensuring the consistency of the stored data in a distributed system.

In addition to traditional integration and validation tests, several applications have been developed to cover the entire range of use for the data storage system. These applications are executed in the frame of scenarios that are representative of present and future earth observation, deep-space and telecommunication missions.

The validation tests performed on the prototype configuration proved the correct behaviour of the system in a distributed environment. Moreover, the performances measured are very good. This demonstrates that GAMMA may be applied to new missions where availability, security and flexibility are key issues.
2.2 Context

The Generic Architecture for Mass Memory Access (GAMMA) study is the logical continuation of the Solid State Mass Memory (SSMM) study leaded by EADS Astrium (formerly Matra Marconi Space) that ended in 2001. The layered architecture defined during the first study has been improved and generalized in order to be independent from the network and storage technologies. Further, scalability is built into the design to achieve the performance required. An implementation on a representative configuration is achieved to validate the architecture and evaluate the performances.

The main requirements of the Generic Architecture for Mass Memory Access are:

- Configurability, portability and technology independence

The GAMMA System Design shall warrant portability over different architectures.

- Integration of COTS

The GAMMA architecture shall support the use of COTS software and eventually hardware.

- Sharing mass memory equipment

Mass memory equipment can have one or more links and can support either point-to-point or packet switching protocols. Several users can be connected to a same mass memory element at a same time. Thus a sharing mechanism has to be implemented to avoid conflicts on resource access and also to ensure data consistency.

- Distributed storage

The overall storage area to manage can be made of one or several discrete mass memory equipment. The entire mass memory system shall be configured as a general resource. It shall be made available to any user either partially or entirely.

- Read and Write

In order to access the data required by a user, it is necessary to define a global addressing method (identification of the data localization).

- Organization

The notion of file types needs a base of ‘rules’ to identify and reference individual files and eventually manage different levels of logical organization.

- Access restriction

The stored data must be protected against forbidden accesses.

- Reconfiguration

The GAMMA system shall cover the entire range of space applications. Some of these applications are very constrained and require a great flexibility in the reconfiguration of the mass memory system.

- Scalability

For the same reason of adaptation to different mission needs, the architecture shall be scalable for capacity and performance.
2.3 Organization

The project was organised into two phases. The first phase aims to define the main requirements of the new mass memory management system with respect to past, present and future mission needs and to propose a new architecture able to cope with these requirements. This phase had to be completed within 8 months. The second phase plans to refine the architecture and implements it on a representative prototype in order to check its correct behaviour and measure its performances. This phase had to be completed within 8 months.

The entire project has been led by EADS Astrium with the participation of three different national entities. EADS Astrium SAS (F) was the prime contractor and had the responsibility of the project management, the definition of the architecture and the realization and validation of the prototype. Thanks to their unique experience in the frame of Observation, Science and Telecommunication missions EADS Astrium GmbH (D) and EADS Astrium Ltd (Uk) worked on the definition on the requirements and on the definition of reference scenarios to be used for the validation of the system.

Phase 1

The main tasks executed during this phase were:

- Requirements Capture and Consolidation
  Analysis of previous study requirements and capture of new requirements from present and future developments

- Specification of User Services and Associated Service Primitives
  Definition of the services to be provided to users (mainly data access and protection).

- System Analysis and Architecture Design Document
  Analysis and definition of the complete and generic data access architecture
  Definition of the reference scenarios representative of future use of the mass memories.
  Definition of the prototype configuration able to validate the new concepts.

Phase 2

The main tasks executed during this phase were:

- System and Software Architecture Detailed Design
  Refinement of the system architecture

- HW/SW Detailed Design and related performance evaluation
  Performance analysis of the system.

- Prototype Implementation
  Implementation on a representative configuration

- Prototype Verification
  Validation of the system by checking its behaviour in a distributed environment and measure of its performance.

- Assessment, Evaluation and Future Directions
  Evaluation of the validation results and proposition for future works.
Figure 1: GAMMA Phase 1

Figure 2: GAMMA Phase 2
3 Phase 1

3.1 Requirements and constraints

This task consisted in the analysis of previous study requirements as well as present and future mission requirements in the entire range of space applications. This covers the capture and the consolidation of the needs in term of data capacity, data rate and services of a set of applications the full range of the space domain: optical and radar observation, telecommunication, science and human in space. If most of the requirements were already identified during previous studies, some new requirements appear and some others are no more relevant. The specific requirements described in the Statement Of Work were also analyzed and refined in system requirements to match configurability, reconfigurability and scalability target.

The analysis of previous study requirements covered the following domains:

- Earth observation (optical)
  - Spot5/Hélios2

- Earth observation and science (radar)
  - ASAR
  - HSRRA

- Deep Space and science
  - Eureca
  - Soho
  - Mars Express
  - Rosetta

The capture of new requirements from present and future developments focused on the following missions:

- Earth observation (optical):
  - Pléiades EADS Astrium SAS & Ltd

- Earth observation and science (radar)
  - TerraSAR – EADS Astrium GmbH
  - CryoSat – EADS Astrium GmbH
  - SAR for export mission – EADS Astrium Ltd

- Man In Space (science)
  - Columbus (new mass memory) – EADS Astrium SAS
Figure 3: Main missions analyzed
For each of the missions the following analyses have been performed:

- **Functional requirements:**
  - Data storage: data organization required, mass memory used by instrument and central processor (HK data).
  - Data processing: access by on-board processors, e.g. for compression or on-board analysis.
  - Data access: data multiplexing and data downlink management, transfer of raw data and packets (CCSDS).
  - Data integrity: manage memory failures, reconfiguration, support detection and autonomous avoidance of bad memory areas.
  - Support flexible redundancy configurations

- **Performance requirements**
  - Data capacity: extremely variable depending on the mission (from few Mbits to hundred of Gbits). A great flexibility is required to save power and increase reliability and maintainability of the system.
  - Data Input/Output flow: extremely variable depending on the mission (few Kbps up to several Gbps), several I/F.

- **Analysis of protocol requirements**
  - Logical organization: manage data I/F, access at several levels: raw data for instruments, complex data organization for processors.
  - Data control: handle asynchronous and asymmetric transfers

Additional Mass memory requirements and constraints defined in the SOW have been included:

- **Configurability, portability and technology independence**
  The GAMMA System Design shall warrant portability over different architectures. This includes different user architectures, different networks and different mass memory technologies.

- **Integration of COTS**
  The GAMMA architecture shall support the use of COTS software and eventually hardware. The main function that can be supported by COTS software is the file management. The mass memory can also include COTS hardware as flash memories or hard disks.

- **Sharing mass memory equipment**
  Mass memory equipment can have one or more links and can support either point-to-point or packet switching protocols. Several users can be connected to a same mass memory equipment at a same time. Thus a sharing mechanism has to be implemented to avoid conflicts on resource access and also to ensure data consistency.

- **Distributed storage**
  The overall storage area to manage can be made of one or several discrete mass memory equipment. The entire mass memory system shall be configured as a general resource. It shall be made available to any user either partially or entirely. It shall be possible to transparently use the entire storage area as a single logical storage unit, or at the opposite, define private areas to dedicated users.
• **Read and Write**

In order to access the data required by a user, it is necessary to define a global addressing method (identification of the data localization).

• **Reference**

Referencing files (group of data) implies that GAMMA shall provide a way to associate the reference (Users identification of the file) with the address of the relevant group of data it represents. This introduces the concept of a ‘File Reference Table’ that contains one entry per declared file and each entry shall at least associate the reference and the address where data are stored. This reference is generally supported by the file management software and can have different implementation (e.g. FAT, I-node, etc.).

• **Organization**

The notion of file types needs a base of ‘rules’ to identify and reference individual files. The reference convention adopted by the overall file management software meets this requirement. The convention shall support several levels of classification and provide a hierarchical view of the file organization. The organization can also be supported by the definition of different logical level of hierarchy, e.g. volumes, partitions and directories.

• **Access restriction**

Access restriction requirement can be covered at several levels:

  o At System level, the access restriction can be implemented by defining reference restrictions at User level: each User can only access files using a specific reference.

  o At File Management System level, the access restriction implies the identification of each User and the definition of access rights to each file or partition.

• **Reconfiguration**

The GAMMA system shall cover the entire range of space applications. Some of these applications are very constrained and require a great flexibility in the reconfiguration of the mass memory system. This reconfiguration concerns the storage capacity with the need to add or remove memory areas, the network with the support of different interfaces and the software with the possibility to replace the data management policy.

• **Scalability**

For the same reason of adaptation to different mission needs, the architecture shall be scalable for capacity and performance. The capacity scalability is obtained by supporting one or several memory equipment with potentially different sizes. The performance scalability is reached by supporting different networks and number of interfaces at the level of each memory equipment. Moreover, the architecture shall support the implementation of the functions at the more optimized level, i.e. software, firmware or hardware level. The localization of the functions shall also be optimized by taking into account the capacity of the system elements.
3.2 User services and primitives

The interfaces to be provided by the mass memory system to its users and the interfaces that must be supported at the hardware level in order to match portability and performance requirements have been identified early in the project to support the co-design approach.

These services are mainly focused on the data management, including the logical organization supported by partitions, directories and files. A complete set of services has been defined in order to cover the requirements previously defined. These services are very similar to the services provided by common file management services and are listed in the following tables.

Table 3.2-1: GAMMA user services

<table>
<thead>
<tr>
<th>Initialization</th>
<th>Partition management</th>
<th>File management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize the FMS library</td>
<td>Define a partition</td>
<td>Create a standard file</td>
</tr>
<tr>
<td>Initialise a partition</td>
<td>Create a packet ring buffer</td>
<td></td>
</tr>
<tr>
<td>Mount a partition</td>
<td>Create a packet linked list</td>
<td></td>
</tr>
<tr>
<td>Unmount a partition</td>
<td>Extend an existing contiguous file</td>
<td></td>
</tr>
<tr>
<td>Check a partition</td>
<td>Delete a file</td>
<td></td>
</tr>
<tr>
<td>Synchronize a partition</td>
<td>Rename a file</td>
<td></td>
</tr>
<tr>
<td>Direct write on a partition</td>
<td>Copy a file</td>
<td></td>
</tr>
<tr>
<td>Direct read of a partition</td>
<td>Move a file</td>
<td></td>
</tr>
<tr>
<td>Retrieve information on a partition</td>
<td>Get file attributes</td>
<td></td>
</tr>
<tr>
<td>Reserve access to a partition</td>
<td>Set file attributes</td>
<td></td>
</tr>
<tr>
<td>Unreserve access of a partition</td>
<td>Get file statistics</td>
<td></td>
</tr>
<tr>
<td>Enlarge a partition</td>
<td>Lock a file</td>
<td></td>
</tr>
<tr>
<td>Shrink a partition</td>
<td>Unlock a file</td>
<td></td>
</tr>
<tr>
<td>Defragment a partition</td>
<td>Directory management</td>
<td></td>
</tr>
<tr>
<td>Move a FMS Partition</td>
<td>Create a directory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remove a directory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rename a directory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Found entries in a directory</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standard data file access</th>
<th>Packet ring buffer access</th>
<th>Packet linked list access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open a file</td>
<td>Open a packet ring buffer</td>
<td>Open a packet linked list</td>
</tr>
<tr>
<td>Close a file</td>
<td>Close a packet ring buffer</td>
<td>Close a packet linked list</td>
</tr>
<tr>
<td>Read data from a file</td>
<td>Retrieve a packet from a packet ring buffer</td>
<td>Retrieve a packet from a packet linked list</td>
</tr>
<tr>
<td>Write data to a file</td>
<td>Read a packet from a packet ring buffer</td>
<td>Read a packet from a packet linked list</td>
</tr>
<tr>
<td>Move file access pointer</td>
<td>Store a packet into a packet ring buffer</td>
<td>Insert a packet into a packet linked list file</td>
</tr>
<tr>
<td>Synchronize a file</td>
<td>Move file access pointer</td>
<td>Remove a packet from a packet linked list</td>
</tr>
</tbody>
</table>
3.3 Reference scenarios

The goal of this task was to define representative scenarios for the use of future mass memories. These scenarios have to cover the entire range of space applications. They are later used to define the validation scenarios of the GAMMA architecture and are implemented and executed on the prototype configuration.

Three main scenarios have been defined to cover Earth observation, Science and Telecommunication missions. Each scenario covers an anticipated range of applications based on a synthesis of the requirements with an emphasis on high performance. The main characteristics of the scenarios are described hereafter.

3.3.1 Earth Observation mission

This scenario is representative of an earth observation mission supported by one spacecraft that includes an optical instrument, e.g. PLEIADES. The reference architecture for this kind of mission is depicted in Figure 4. The requirements concerning the mass memory for this kind of mission are:

- Operational Modes: Write only; Read only; Simultaneous Write+Read
- Periodic high data rate operation (read/write) around a LEO orbit of typical duration of 100 minutes.
- Data integrity support: SEE protection (EDAC and memory scrubbing).
- The number of active channels, the mean data output rate per channel and packet size can change according to the mode selected.
- Data rates and volumes can vary within a mode depending on the scene.
- CCSDS Source Packets of varying sizes (e.g. from 100 to 64k octets).
- Instrument contains very limited data buffering and expects to be able to send its measurement data immediately.

- Key parameters:
  - Storage Capacity (EOL): 600 Gbits
  - Total data input rate: Up to 1.8 Gbps
  - Number of active input channels: Up to 6 simultaneously
  - Highest mean input rate per channel: 900 Mbps
  - Total data output rate: 620 Mbps
  - Number of active output channels: Up to 4 simultaneously
  - Downlink Data output rate per channel: 155 Mbps

These requirements are very exacting in term of capacity and data rates, but data organization remains simple.

The scenario defines a typical use case with two hypotheses: a higher priority downlink for new measurement and the simultaneous observation and downlink. This scenario is illustrated in figure Figure 5.
At the start of the G.S. contact period, begin downlink of stored measurement data and associated auxiliary data.

Begin acquisition of new measurement data from instrument and new aux data from CDMU.

Switch from downlinking previously stored data to downlinking the newly acquired data.

At the end of the new acquisition, and when all the new acquisition measurement data has been downlinked, return to downlinking previously stored data until the end of the contact period.

**Figure 4: Earth Observation reference architecture**

**Figure 5: Earth Observation reference scenario**
3.3.2 Deep Space mission

This scenario is representative of a deep space mission supported by one spacecraft that includes several instruments, e.g. Mars Express. The reference architecture for this kind of mission is depicted in figure Figure 6. The requirements concerning the mass memory for this kind of mission are:

- Storage of instrument measurement data packets prior to the downlink of this data to ground.
- Storage of telemetry data packets from other sources which also go to make up the complete downlink data stream: status data from instruments and platform, relayed probe data, visual telemetry data, etc.
- Storage and retrieval of data which is not part of the downlink data: instrument parameter files, command files and timelines, software (e.g. executable code), housekeeping data files, etc.
- Compression of data using various algorithms e.g. Wavelet algorithm and CCSDS/Rice algorithm (for lossless compression).
- Sending of stored telemetry packets to the downlink data formatting (transfer frame generator) and transmission facility.
- Storage to support processing of image data for navigation.
- Extended memory resource for processing modules, e.g. use mass memory as swap area.
- Flexible multi-user storage system capability.
- Has to accommodate a mixture of concurrent accesses by the range of users.
- Operational Modes: Write only; Read only; Simultaneous Write+Read.
- Data integrity support: fault detection and automatic copy of critical files to independent Mass Memory, data mirroring service option, EDAC service for memory data access, Memory data scrubbing.
- Different types of storage are used including packet handling with the following characteristics: simple data sequence used for storing data in the SSMM and for reading it from the SSMM for downlinking, multiple cyclic buffers operating in parallel, both FIFO and LIFO operation.
- Needs to allow user-controlled random access to data files for other types of data
- CCSDS Source Packets (of varying sizes up to 64k octets) and plain files (of varying sizes up to the maximum capacity of the SSMM).

Key Parameters:
- Storage Capacity (EOL): 25 Gbits
- Total data input rate: Up to 40 Mbps
- Number of active input channels: Up to 9
- Highest mean input rate per channel: 20 Mbps
- Number of active output channels: Up to 9
- Downlink Data output rate: 230 kbps

These requirements are very limited in term of capacity and data rates, but data organization and resource sharing are very important.

The scenario defines a typical use case with several hypotheses and is illustrated in figure Figure 7.
At the start of the measurement period, commence measurement timelines and initialise instruments.

Switch instruments into their operational modes.
Downlink measurement data.

Perform short burst(s) of measurements at the highest data input rate.
Downlink measurement data.

Switch instruments out of their operational modes.
Downlink measurement data.

At the end of the measurement period, revert to standby state - complete measurement data downlinking.

Figure 6: Deep Space reference architecture

Figure 7: Deep Space reference scenario
This scenario should be representative of a future telecommunication mission supported by one spacecraft that includes one complex telecommunication payload, e.g. Intelsat X. Presently, there is mass memory in telecommunication space systems. The reference architecture for this kind of mission is depicted in Figure 8. The requirements concerning the mass memory for this kind of mission are:

- Flexible multi-user storage system capability.
- Has to accommodate a mixture of concurrent accesses by the range of users.
- Operational Modes: Write only; Read only; Simultaneous Write+Read.
- Data integrity support:
  - Fault detection and automatic copy of critical files to independent Mass Memory.
  - Data mirroring service option.
  - EDAC service for memory data access.
  - Memory data scrubbing.
- Key Parameters:
  - Storage Capacity (EOL): 10 Gbits
  - Number of active input channels: Up to 5
  - Highest mean input rate per channel: 10 Mbps
  - Number of active output channels: Up to 5
  - Highest mean output rate per channel: 100 Mbps

These requirements are very limited in terms of capacity and data organization but the transfer from the mass memory system to the payload must be minimized, i.e., the data rate must be maximized.

The scenario defines a typical use case is illustrated in Figure 9.
Figure 8: Telecommunication reference architecture

Receive configuration data (low data rate)
Store data in areas dedicated to each DSP.

Send configuration data to each DSP at maximum data rate.

Figure 9: Telecommunication reference scenario
3.4 System Analysis and Architecture

A system analysis has been performed on the basis of the results of the first SSMM study ([SSMM ADD] and [SSMM DDD]). It also took into account the definition of the system objectives and requirements defined in [GAMMA RSD] and the interfaces described in [GAMMA USP]. The results of this analysis have led to define a new architecture.

The entire GAMMA architecture has been modelled using UML with the support of the free POSEIDON tool in its version 2.6. More recent versions of these tools exist but are no more compatible with a commercial use (for the free version).

The next figure shows the environment of the mass memory in a space system. This clearly show its criticality as it can be used concurrently by both Payload and Platform.

**Figure 10: Mass memory environment**

The mass memory itself is depicted in the next figure.

**Figure 11: Mass memory physical organization**

The mass memory subsystem is made of one or more memory equipment. The memory equipment supports a storage capability. The storage area is made of sectors. The size and the number of sectors may vary from a storage area to another. Solid-State Mass Memory (SSMM), SafeGuard Data Recorder and Hard disk are all memory equipment.
The network supporting the exchange of data and commands between the mass memory and the different users is shown in the next figure.

Figure 12: Physical network

The physical network is made of at least two interfaces and one link. Routers are optional. Each link connects two interfaces. A router provides 2 or more interfaces. All network standards support the emission and the reception of data packets.

All LAN 802.3, SpaceWire and MIL-STD 1553B networks are compliant to this definition. The LAN 802.3 is a packet switched network with optional routers. The SpaceWire network is either a collection of point-to-point links or can include static or dynamic routers. The MIL-STD 1553B network has a bus topology and does not include any router. The next figure illustrates the connections between the users of the mass memory system and the mass memory equipment.

Figure 13: Mass memory system network

Each user can be connected to any number of each different network. This means that a single user can be connected to one or several networks of a same type (redundant networks). It can also be connected to different networks, e.g. one LAN 802.3 network and a SpaceWire network. On the other side, the memory equipment connections respect the same rules.
Because the mass memory function can be supported by several mass memory units and because it can also be concurrently used by several users within the system, a particular analysis has been performed to ensure the correct data and transaction protections.

This analysis has led to define some major design features:

- The Partition layout mechanism supports the decomposition of a logical partition into several physical memory areas. A logical partition is always seen as a contiguous area from user point of view (e.g., COTS FMS libraries). This mechanism increases the safety and reliability of the mass memory system and eases its reconfiguration.

- To reach flight requirements, a protection layer (wrapper) must surround the COTS components. The wrapper filters the inputs and outputs of the component, so as to:
  - Adapt the interface according to system needs,
  - Prevent illegal requests that could lead to unsafe behaviour of the component,
  - Detect erroneous outputs from the component and prevent their propagation.

The wrapper is illustrated in the next figure. One wrapper is developed for each COTS library. These wrappers are included in the FMSLibrary component.

---

**Figure 14: Partition layout**

**Figure 15: The wrapper**
The distributed layered architecture that is a direct evolution of the layered architecture defined during the SSMM study. It has been simplified and generalized to match the new requirements and especially to be more generic, configurable and scalable. Several layers implement different levels of services from communication to file management. A service provided by a layer can be implemented on any component either in software or hardware. The next figure presents an overview of the architectural design.

**Figure 16: The Generic Architecture for Mass Memory Access**

The functions of the various software layers are as follows:

- **a.** The API Services layer provides all the interfaces for Application Programs to make use of the FMS functions. This layer allows defining a standard interface to access the mass memory system whatever its hardware and software implementation.

- **b.** The FMS Services layer provides the implementation of the File Management Services. This layer can easily include a standard COTS file management software or a specific one.

- **c.** The MM FMS Services layer provides an interface between the FMS required interfaces and services provided by the mass memory primitives. It can be seen as a FMS Services driver that interfaces with the SSMM peripheral devices.

- **d.** The MM Primitives layer provides an interface to the mass memory management operations and implements the low-level operations and communication protocol management that operate the mass memory Hardware.

- **e.** The Connection Services layer allows for the exchange of information between users and memory modules.

The layered architecture supports an easy adaptation to system components ensuring a correct access to the storage are through complex software libraries or very simple hardwired access primitives.

**Figure 17: GAMMA architecture adaptability.**
The following figure presents an overview of the GAMMA architecture using a class diagram representation. The operations provided by each class are not shown here. The implementation of the operations is independent from the hardware. Each operation can be indifferently implemented on any user or memory equipment hardware.

Figure 18: General view of the layered architecture

This representation clearly shows the decomposition of the system into layers. Each layer uses the underlying layer. In some cases, the applications can have a direct access to the layers that are not at the immediate inferior level. One example concerns the configuration of the network during the initialisation of the system.

Once the architecture has been defined, a prototype configuration has been proposed with the objective to validate the correct coverage of the requirements and measure the performances. This prototype also needs to be representative of the architecture of future systems. Moreover, it has to be able to support the execution of both Software and Hardware functions. Finally, it has to be sufficiently configurable to make possible the definition of several architectures to match scenario requirements.

For these reasons, a configuration based on five identical FPGA boards and a SpaceWire network has been proposed. Each board is equipped with a six million gates FPGA, able to support the programming of a LEON2 processor and specific hardwired functions (SpaceWire IP cores and GAMMA hardwired services), and 512 Mbytes of SDRAM used either for software execution or for supporting the mass memory storage. All the boards being identical, each board can be configured either as a memory user (LEON2 processor running Software) or as a memory module supporting data storage.
4 Phase 2

4.1 System and Software Architecture Detailed Design

The first task of phase 2 consisted in the refinement of the system architecture. This includes the description of the activities of each service and the analysis of the dynamic aspects with the use of Message Sequence Charts (MSC). The detailed design also focused on the main mechanisms as the management of virtual memory, the protection of the data, the access to shared memory areas and the design of event and error services. Most of the detailed design of the GAMMA architecture has been done using POSEIDON, some parts have been designed using UML formalism but in manual form.

As an example, the activity of the memory protection services called areaLock is given in the next figure.

Figure 19: areaLock activities
The event services depicted in the following figure support the transmission and management of asynchronous messages within the system. The events are processed at the most appropriate level and only the main failures are propagated to the system FDIR application. Moreover, the event registering mechanism makes possible and easy the insertion of new layers or the interception of the events.

![Diagram of event management](image)

**Figure 20: The event management**

### 4.2 HW/SW Detailed Design and related performance evaluation

This task focused on the analysis and evaluation of the performance of the system taking into account its architecture and the prototype configuration defined at the end of phase 1. This configuration relies on LEON processors for users and a SpaceWire network. The performance evaluation has been performed on the basis of the reference scenarios also defined during the first phase (see chapter 3.3).

Taking into account the processor capacity, an evaluation of the complexity of both software and hardware functions as well as the structure of the SpaceWire network, the evaluation of the performance of the GAMMA system showed that it was able to support the data rate and functions required by validation scenarios.
4.3 Prototype Implementation

To validate the new architecture and concepts, an instantiation of GAMMA has been realized on a representative prototype configuration. This implementation has been used to execute the validation tests and to measure the performances of the system.

4.3.1 GAMMA prototype elements

GAMMA is a generic architecture able to cope with different architectures and networks. To validate this architecture, it is necessary to perform an implementation. This implementation has been made on a prototype made of representative elements for memory users, memory boards and network.

4.3.1.1 Memory users and modules

For both memory users and memory boards, a FPGA board has been selected. This board make possible the implementation of existing or new IP cores. Moreover, by providing five identical FPGA boards, the GAMMA prototype becomes highly configurable in terms on number of memory users and memory modules. Actually, the prototype can be configured to have from one memory user and four memory modules to four memory users and one memory module. An FPGA board is shown in the next figure.

![FPGA board](image)

**Figure 21: The FPGA board used to implement memory users and memory modules**

All boards are equipped with 512 Mbytes of SDRAM memory (4 Gbits). The use of identical memory boards also greatly eased the integration of the prototype and makes possible different configuration because any board can be configured either as a memory user or as a memory board.

4.3.1.2 Network

It exists numerous bus and network standards for space applications. The more common networks are based on MIL-STD1553, CAN and SpaceWire standards. In order to validate the architecture for a complete set of applications and in particular for applications requiring high data rate accesses, the SpaceWire network has been retained to implement the network of the prototype.

The SpaceWire network is made of three distinct elements. The SpaceWire IP core is implemented on all memory users and memory modules. A SpaceWire router supports the packet routing function. The cables establish the connections between all the elements.

Several SpaceWire IP cores are available on the market. Two of them are distributed by ESA. One is coming from University of Dundee (UoD) and the other is coming from EADS Astrium Vélizy. This is this last IP core that has been selected for the GAMMA implementation because it is delivered in two versions, one standalone version and another version ready to be connected to an AMBA bus. The GAMMA prototype needs these two versions because memory users include a LEON processor and the SpaceWire IP core is connected to its AMBA bus while in memory users the SpaceWire IP cores are connected to the GAMMA Primitives. Moreover, the DCM core from Xilinx has been used to increase the frequency of the SpaceWire IP cores up to 100 MHz.
Several SpaceWire routers are also available on the market, all are expansive. The SpaceWire router used in the frame of the study has been lent by ESA. This is the router made by StarDundee. It provides height SpaceWire ports and one USB port to connect a PC running Windows. The router is shown in the next figure.

![Figure 22: The SpaceWire router](image)

To be able to connect the FPGA boards to the SpaceWire network, an interface board with two DB9 connectors and two SpaceWire standard connectors has been developed especially for the GAMMA prototype. This interface board is plugged on the FPGA board as a mezzanine on the generic IO connector. The two DB9 connectors are used to connect the serial interfaces of the LEON processor. The interface board is shown in the next picture.

![Figure 23: The 2 RS232/2 SpaceWire interface board](image)
4.3.2 GAMMA prototype configuration

The GAMMA prototype configuration used for the validation is the default configuration, i.e. the delivered configuration. It includes three boards configured as memory user and two boards configured as memory board. One memory user has two SpaceWire links as the two other memory users have a single SpaceWire link connected. The two memory boards have two links connected.

The complete GAMMA prototype configuration and his parts are described in the GAMMA configuration item data list document [GAMMA CIDL]. It assembly is described in [GAMMA UM]. It is depicted in the next figure.

![Figure 24: GAMMA prototype configuration used for validation](image)

The picture of the final prototype delivered to ESA/ESTEC is shown hereafter.

![Figure 25: GAMMA prototype configuration assembled](image)
4.3.3 GAMMA prototype implementation

The GAMMA architecture has been instantiated to run on the prototype configuration previously described. On mass memory user boards, all the layers have been developed in Software except for the communication layer that also includes hardwired functions (SpaceWire and CTM). The memory boards only implements the communication and primitives layers. Two versions of memory boards have been developed. One version where the two layers are implemented by software and one version entirely developed with hardwired functions. The Software version has been used to validate the mass memory primitives services and during the first integration steps of the memory users. This version gives a great visibility on operations. Once a memory primitive is checked in its software version, it is developed in its hardware version. Then, the hardware version is connected to the memory users and its behaviour is checked. In order to keep the consistency of both software and hardware versions, all the modifications performed on one side is carried forward on the other side.

![GAMMA implementations](image)

Figure 4-26: GAMMA implementations

The configuration of target boards FPGA differs from memory users to memory modules.

- The memory user boards are programmed with a LEON2 processors, two SpaceWire interfaces and the CCSSD Time Manager that is used for test purpose only. The user boards execute all GAMMA software functions of the different layers on top of a real-time operating system. The mass memory primitives layer only contains a stub to the services that are implemented on the memory board side. The FMS layer includes three FMS libraries: one raw data access library, one CCSDS packet stack library and one DOS-like (FAT based) COTS library called ERTFS.

- The Software memory module boards have exactly the same configuration than the memory user boards. The only difference consists in the modification of the software that runs on the LEON2 processor. The MM primitives are not stubs but real implementation of the mass memory primitives functions. The data storage is ensured by direct access to the memory of the boards through pointers.

- The Hardware memory board configuration also includes a LEON2 processor but it is not used by primitives and it does not include the PCI interface. Presently, it has been included only to access its SDRAM memory controller through the AMBA bus. This has been considered as the only possible way to be able to develop the memory modules with respect to the budget and timing of the study. The memory module includes the GAMMA primitives that are connected between the SpaceWire interfaces and the AMBA bus to access the memory. The configuration of the memory module is illustrated in the next figure.
The GAMMA primitives are developed as a building block that is integrated in the configuration of a SOC design. It is directly connected to the SpaceWire interface (SpW IP core) to receive commands and to send back results. It is also connected to the SDRAM memory controller that is used to access the memory using a burst mode.

Two instances of the SpaceWire interface and Primitive Services are implemented in the same FPGA. They can be activated concurrently and are then protected by means of semaphores because they use some common services and protection tables. This configuration is depicted in the next figure.
As previously indicated, any target board can be configured as a software memory user, as a software memory module or as a hardware memory module. In order to keep the consistency between hardware and software memory modules, the memory mapping has been made similar between all boards as illustrated in the next figure.

![Memory Mapping Diagram](image)

**Figure 29: Target boards memory mapping**

With this configuration, it is possible to implement software memory module by implementing primitives on top of the RTOS with data storage occupying the same address range than hardware memory modules. Moreover, it is possible to dump the entire content of a hardware memory module in the unused memory area of a user.

### 4.4 Prototype Verification

This task aims to validate the system by checking its behaviour in a distributed environment and measure of its performance. The verification is based on specific robustness and performance tests as well as the execution of reference scenarios.

#### 4.4.1 Functional results

All the tests passed on the GAMMA configuration at the level of each layer are successful. The GAMMA configuration respects its specification. It supports the access and management of data stored in mass memory module using different data or file management library. The concurrent accesses are correctly managed and the consistency of the stored data is ensured at any time.

#### 4.4.2 Performance results

The performance of the system has been measured at different level. The GAMMA prototype is able to support up to 100 Mbps on SpaceWire links when no software is used. The next table gives results for this test.

<table>
<thead>
<tr>
<th></th>
<th>link 1 to link 2</th>
<th>link 2 to link 1</th>
<th>link 1 to link 2 &amp; link 2 to link 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>transmission rate</td>
<td>99.9424</td>
<td>99.5328</td>
<td>47,816</td>
</tr>
<tr>
<td>global board rate</td>
<td>199,8848</td>
<td>199,0656</td>
<td>266,0352</td>
</tr>
</tbody>
</table>

**Figure 30: SpaceWire communication data rates**
As soon as software is used, the data rate is limited by the frequency (40 MHz) of the FPGA board supporting the LEON2 and the AMBA bus to transfer the data, the software drivers performing a memory copy to build packets containing the commands. Taking into consideration this limit imposed by the prototype configuration, the data rates that can be reached by the memory users are good.

A single memory user accessing a memory module at the level of the Primitives layer has a read data rate that exceeds 71 Mbps while the write data rate approaches 58 Mbps. The maximum write data rate remains unchanged when the two links of the memory module are simultaneously used, while the maximum read data rate slightly decreases due to the limitation of the bandwidth supported by the memory module. This limitation is due to the implementation that relies on the memory controller of the LEON2 processor. A dedicated memory controller should support higher data rates. The next figure summarizes the data rate measured at the level of the primitives.

![Memory Module Data Rates](image)

**Figure 31: Memory module data rates**

When the memory users access the storage area at the level of the FMS layer, the data rates obtained are decreased. This is due to the multiple accesses required by the file management libraries to maintain the data structures. Moreover, the maximum data rates that can be reached are tightly linked to the size of data exchanged with the mass memory. Larger exchanges decrease the overhead due to the protocol and file management operations. It appears that it is totally unsuitable to read or write data having a size smaller than the size of a memory sector (512 bytes in GAMMA configuration). Nevertheless, the maximum data rates obtained with largest data exchanges remain interesting. The read data rate is typically 30 Mbps and the write data rate is 26 Mbps. It should be noticed that the GAMMA Software has been configured to ease the tests and the debugging and not to optimize its performances. The next figures summarize the data rates obtained with the three different file systems.
A summary of the maximum data rates that can be reached on the prototype configuration is illustrated in the next figure.

Figure 32: ERTFS and PRBFS performance (one or two users accessing a memory module)

Figure 33: RAWFS and PRBFS performance (one or two users accessing a memory module)
In addition, the CPU usage of the primitive has been evaluated to 61.8% for read operation and 52% for write operation at full data rate. This means that respectively 38.2% and 48% of the time of a transaction is used by the communication and processing by memory modules. This lets some place for increase of SpaceWire communication data rate and improvement of hardware memory primitives.

An illustration of the performance measures is given in the next figure.

The tests and measures performed on the GAMMA prototype configuration also show a degradation of the data rate of read operations when several applications access a same memory module. This degradation is due to both SpaceWire and Memory Module primitives. Actually, a link receiving and executing a memory access command is considered as free from the SpaceWire router. So the next command in destination to that memory module is routed, by default to the same link, even if the other link is not used. The last command remains blocked until the end of the execution of the previous one. This is particularly clear for the read operation for which the command is very small and the response can be long. This observation should be taken into account for the performance estimation of future systems.
4.5 Assessment, Evaluation and Future Directions

This chapter presents an assessment of the work performed during the GAMMA study and of the results obtained on the prototype.

4.5.1 Overall study assessment

In the GAMMA distributed layered architecture, the services can be implemented by hardware or software. This decomposition has been led by technical and performance considerations with the support of a co-design approach. This approach has made possible the quick development and integration of the prototype following the process depicted hereafter.

The selection of the services to develop in hardware has been made following different considerations. The main consideration was the complexity of the service and the need for flexibility. Another consideration was the limited time and budget of the study. After analysis, the decomposition between hardware and software has been defined at the level of Primitives layer. On memory modules, the primitives are entirely developed in hardware using a high-level language (HandelC). On memory users, the primitives layer only contains stub services. This decomposition is illustrated in the next figure. The blue elements are hardware while green elements are software.

![Figure 36: The co-design process](image)

![Figure 37: Hardware/Software decomposition](image)
This decomposition has made possible the development of memory modules that only include hardwired services. This approach should ease the development of a future mass memory systems based on GAMMA services. The complex operations as FMS management can be difficult to implement in hardware.

The co-design process has been followed for the development of GAMMA Primitives layer. During this process, the primitives layer services have been developed in both software and hardware. This parallel development has numerous advantages:

- It make possible to have an operational prototype even if hardware development failed, based on memory modules including Software Primitives.
- It make possible to validate the behaviour of the Primitives services before their development in hardware.
- It is a good support for discussion between hardware and software development teams.
- It make possible the test and integration of the higher level layers (Application, FMS and MM FMS) very early.

The co-design process has been supported by different tools. The Magillem Prosilog tool supported the design of both memory user and memory module systems. Despite some bugs identified during its use, the application has been a great help to quickly build the configuration of the two systems. These systems are built by including and connecting blocks coming from different sources (Magillem existing blocks, external blocks as SpaceWire IP cores and specific blocks as Primitives).

Several iterations between hardware and software versions of the primitives have been performed. These iterations are typical in rapid prototyping approaches. In GAMMA they made possible the verification of services in software before their implementation in hardware leading to time saving because software verification is much quicker than hardware verification. Moreover, the software version of the primitives has also been modified to be as representative as possible to hardware implementation. At the end of the process, the two versions of the primitives are similar in term of provided services and behaviour. During nearly all the development phase, a software version of the primitives were available and used to develop, test and integrate higher-level layers of the architecture.

Of course, this approach and process has been greatly optimized by the fact that hardware and software development teams were at the same premises, in two adjacent desk rooms and very open to discussion. Thanks to this proximity, the time required for each iteration was very short, sometimes less than one day. The next figure summarizes the development planning and clearly shows that Hardware and Software activities have been performed in parallel.

![Co-design planning diagram](image)

Figure 38: Co-design planning
4.5.2 Functional assessment

The validation tests performed at all levels in the GAMMA architecture are successful. All the services perform correctly their function. Only two minor problems have been identified in the use of the ERTFS COTS library.

In any case, the present GAMMA configuration is operational and can cover a large range of applications in term of supported services as shown by validation scenarios. Moreover, the decomposition into different layers make possible the development of large applications accessing the mass memory through a file management library as well as the development on instrument accessing the mass memory at the level of MM FMS Services or Primitives layer. In addition, the protocol used to access the memory module being extremely simple, the connection of real payload modules fully implemented in hardware can be foreseen. These different configurations are depicted in the next figure.

![GAMMA configurations](image)

Figure 39: GAMMA configurations

4.5.3 Performance assessment

The performances obtained on the GAMMA prototype are very good with respect to its configuration. Thanks to the Xilinx DCM IP core, the frequency of the SpaceWire IP cores has been increased from 40 MHz to 100 MHz. At this frequency, the SpaceWire communication between two FPGA boards reaches the maximum data rate that is 100 Mbps.

The FPGA technology does not allow running the LEON2 IP core at higher frequency than 40 MHz. This frequency limitation is imposed to the LEON2 and its AMBA bus that is in charge to read and write the memory through the SDRAM controller. This controller is used in both memory user and memory module configurations. However, on memory modules, the access to the controller is optimized by means of the use of a “burst” mode. The data rate and CPU time measures performed during the validation clearly show that limitation. The maximum data rates obtained at the level of memory module primitives reached nearly 90 Mbps for reading and 73 Mbps for writing. These data rates are representative of data rates that can be reached by instruments or payloads that directly access the memory module. When memory users access the memory modules through the FMS level, the data rates reaches 46 Mbps for reading and 42 Mbps for writing. These data rates are representative of complex CDMU or Processor modules requiring a flexible organization of the mass memory areas.

The CPU times measured for read and write operations clearly show that margins for performance increase exist on the prototype configuration. These margins are at the level of the SpaceWire network that could be used at higher frequency and especially at the level of the memory module SDRAM controller. Actually, the data transfers have been limited to 64 Kbytes and data rate measure diagrams show that larger data transfers could increase the maximum data rate. However, these diagrams also show an inflection of the curves and the increase should be limited to only few megabits per second. In any case, the increase of packet size will generate higher latencies in the system when more than two users access a same memory module.

For same predictability concern, the management of read operation should be modified and optimized.
4.5.4 GAMMA future directions

The assessment of the study opens a large perspective for future activities based on the GAMMA architecture and prototype implementation. These future directions are detailed in the next sections.

4.5.4.1 Extension and optimization

As identified by the validation results and assessment, the GAMMA prototype can be extended and optimized. The extensions shall make possible the support of new functions and services, mainly at the level of primitives. The optimizations shall make possible to reach the limit of the performance of the GAMMA architecture.

1. Add new simple HW primitives to minimize the work to perform on Software users and add new functions to hardware users.

Develop data organization management HW primitives to manage and optimize the memory use with the possible support of defragmentation or fragmentation avoidance services.

2. Support storage of CCSDS packet at memory module side under the form of lists or stacks that could be used by any user on the system.

3. Develop a HW/SW memory module in order to ease the development of hardware primitives and have a unique target board that can be used with hardware or software services.

4. Optimize memory module architecture with the integration of a dedicated memory controller that is a reason of the performance limitation of present GAMMA prototype.

5. Increase SpaceWire data rate to exceed 100 Mbps.

6. Port GAMMA Software on board equipped with AT697 to check the compatibility and measure the performance of the GAMMA architecture and software with LEON ASIC running at 100 MHz.

7. Investigate solutions to ensure the predictability of read commands that can be routed by the SpaceWire router to a memory channel already used by another user.

8. Add fault detection mechanisms and generation of events in primitives at the level of memory modules.

9. Increase reliability and availability by supporting different technologies as RAID (level 1 or 5) with the possible use of SpaceWire multicast feature.

10. Include other COTS FMS libraries.

4.5.4.2 FDIR Assessment

The GAMMA study focused on the definition of a new architecture to access mass memories and its implementation for validation purpose. The implementation and results obtained on the prototype confirm the interest of the approach and the use of such architecture into real space missions can be envisaged. However, a clear FDIR policy requires to be defined for the mass memory system. The GAMMA architecture supports all the services to generate, route and process asynchronous events as failure detections but the FDIR policy has been left to the responsibility of the application level. To define the FDIR policy, it is necessary to identify the following items:

- How failures within memory modules are identified?
- How to detect silent failures?
- Which actions to perform in case of failure?
- What kind of reconfiguration operations must be supported?
- Which system component is in charge of reconfiguration?
4.5.4.3 Integration in validation facilities

The integration steps of on-board software rely on the use of different test beds that need to be representative of the final configuration. These test beds can include software simulation of the environment and equipments. This includes the mass memory equipment. To increase the representativeness of the test bed; it is possible to replace software simulations by real equipment or prototypes. The GAMMA prototype is a good candidate to support the mass memory access function of future test beds. Thanks to its configurability and scalability, it can be adapted to various mission needs.

In such test bed, the mass memory could be accessed either through all layers, including the FMS Services layer or by direct use of mass memory primitives or SpaceWire communication. A dedicated memory user should also be configured as a router or multiplexer to concentrate data coming from different users generating low data rates and store them in the GAMMA memory modules.

4.5.4.4 Technology transfer

The GAMMA prototypes proved the interest and capability of the proposed distributed layered architecture. This architecture could be proposed to different missions that are currently in preliminary phases. The specific needs of the missions, especially in the frame of the payload development, could be compared with GAMMA characteristics.

This work could make possible the application of GAMMA concepts to constrained missions as SOLO where different payloads have to share a common mass memory resource. It will also consolidate the GAMMA user requirements defined in the frame of the first phase of the study.

A study on distribution systems for payload data systems called “DisCo” (ESTEC contract 18787/04/NL/JA ) is presently in progress under EADS Astrium responsibility. The objectives of this project are twofold: on the first hand the study shall result in the availability of a middleware product specific to the space domain; on the other hand a reference application fitting the needs of a complex payload requiring distributed processing shall be developed and used to evaluate and refine the space-oriented middleware. In this architecture, the mass memory storage can also be distributed and could take advantage of the GAMMA architecture. The use of GAMMA within the DisCo study or as a future extension is currently assessed.

EADS Astrium has a unique experience in the frame of the mass memory equipment for space applications. Mass memories for Cluster (Science), Mars Express (Deep-space/Observation), CryoSat (Earth Observation) and Pléiades (Earth-observation) have been developed at Astrium. These projects clearly showed that mass memory systems can be critical. It can be interesting to confront the experience of the team developing mass memories to the team who developed the GAMMA prototype. Some improvements in both current mass memory products and GAMMA prototype can be expected.

Other projects and studies that can be concerned by GAMMA results need to be identified.
5 Conclusion

GAMMA proposes a novel approach to match the requirements of future mass memory systems. Its distributed layered architecture supports a great flexibility in its implementation and makes easier and safer the integration of COTS libraries. This architecture is also particularly configurable and can be adapted to specific mission requirements.

The data protection mechanisms ensure the consistency of the data in distributed systems where independent users can concurrently access same memory areas. The virtual memory manager makes easier the management of the memory area and in particular its reconfiguration. It gives a great flexibility on the size of logical partitions accessed by users that can help to reduce the power consumption of the mass memory system. Moreover, the virtual memory manager makes possible the use of standard COTS libraries requiring contiguous memory areas.

The design and development of the GAMMA prototype has been supported by a co-design process that helped to optimize the system and to ease its integration and validation.

The GAMMA architecture has been implemented on a representative platform based on five identical FPGA boards with 512 Mbytes of memory. A SpaceWire network with one router connects the five boards with one or two interfaces. Each FPGA can be programmed with a LEON2 processor or with specific hardwired function and can act as a user or a memory module. The boards programmed with a LEON2 processor execute an RTOS and support the execution of concurrent applications. The memory primitives managing the access and the protection of data have been implemented by hardwired functions, each memory module being able to manage two SpaceWire interface.

Tested with three LEON2 target boards running several tasks accessing simultaneously to two memory boards providing two SpaceWire interfaces, GAMMA has proved its capacity to handle distributed memory systems by ensuring the consistency of stored data and providing good performances with respect to the prototype configuration.

The work carried out in GAMMA Phase 2 led to the development of a complete distributed memory management system. The core components rely on specific Software mechanisms and hardwired functions devoted to manage storage areas and ensure the data consistency of stored data, supporting concurrent accesses and real-time behaviour in a distributed environment.

This work is the starting point for the development of a new generation of mass memory systems for space applications. It demonstrates the benefits of proposed architecture and solutions in this context and calls for further development. It also shows the significant interest of co-design process and co-simulation-based approaches for the development of services in Software and Hardware and their validation. This work should be continued, possibly targeting specific space applications. The results obtained are very promising and the development of additional services should be very attractive for the space industry.