TRP activity

Building Blocks for System on a Chip

Call-off order No.3

Development of a Spacecraft Controller on a Chip

FINAL REPORT

EUROPEAN SPACE AGENCY CONTRACT REPORT

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

ESTEC Contract No. 13345/98/NL/FM

Prepared by : Marc LEFEBVRE
EADS Astrium France
37, avenue Louis Bréguet - B.P. 1
F-78146 Vélizy Villacoublay Cedex

R&D-SOC-NT-338-V-ASTR
Issue 0 Rev 1

ESA STUDY CONTRACT REPORT			
No ESA study Contract Report will be accepted unless this sheet is inserted at the beginning of each volume of the Report			
* ESA Contract N° 13345/98/NL/FM	SUBJECT **Building Blocks for System on a Chip - COO3** **Final Report**		NAME OF CONTRACTOR **EADS Astrium**
* ESA CR ( ) N°	* STAR CODE	No of Volumes : 1 This is volume n°1	CONTRACTOR'S REFERENCE : R&D-SOC-NT-338-V-ASTR Issue 0 Rev 1

ABSTRACT:

The work of COO3 was necessary to evaluate very high integration levels of microelectronics coming with the increasing demand of performance in terms of processing, power and mass.

Increasing amount of functionalities integrated into the same ASIC implies that the development of a whole system relies on the use of pre-developed and pre-tested IP functions. The end target is to be able to construct a whole system by “simply” integrating IP blocks

The goal of the Call of Order 3 “Development of a Spacecraft Controller on a Chip” (SCoC) of the Frame Contract “Building Blocks for System on a Chip” is to demonstrate the feasibility and to evaluate the methodology and development complexity - of a System-On-Chip (SOC) device composed of IP blocks coming from different sources.

The requirements and functionalities of the SCoC come from the previous Call of Order 2. It was then decided to implement a demonstrator of this SCoC into a commercial reprogrammable FPGA mounted on a Compact-PCI board, making the design available for a validation of the performances and functionalities.

Gate level design methodology currently used for ASIC development has to shift one step over to reach a state of block level design methodology, focusing on the validation of the whole system design.

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

Names of authors:

Marc LEFEBVRE (EADS Astrium)

ESA STUDY MANAGER	ESA BUDGET HEADING	
Mr Roland WEIGAND DIV: TOS-ESM		

*Sectors to be completed by ESA

Title : ESA 13345/#3 : Building blocks for system on a chip

SCoC

FINAL REPORT

	Name and Function	Date	Signature
Prepared by	**Marc LEFEBVRE** ***SCoC Design Manager***	10/5/2004	
Verified by	**Jean-François COLDEFY** ***Digital Electronic Expert***	10/5/2004	
Approved by			
Authorized by	**Marc SOUYRI** ***ASIC/FPGA Design Expert***	11/05/2004	
Application authorized by			

Document type	Nb WBS	Keywords

SUMMARY

The work of COO3 was necessary to evaluate very high integration levels of microelectronics coming with the increasing demand of performance in terms of processing, power and mass.

Increasing amount of functionalities integrated into the same ASIC implies that the development of a whole system relies on the use of pre-developed and pre-tested IP functions. The end target is to be able to construct a whole system by “simply” integrating IP blocks

The goal of the Call of Order 3 “Development of a Spacecraft Controller on a Chip” (SCoC) of the Frame Contract “Building Blocks for System on a Chip” is to demonstrate the feasibility and to evaluate the methodology and development complexity - of a System-On-Chip (SOC) device composed of IP blocks coming from different sources.

The requirements and functionalities of the SCoC come from the previous Call of Order 2. It was then decided to implement a demonstrator of this SCoC into a commercial reprogrammable FPGA mounted on a Compact-PCI board, making the design available for a validation of the performances and functionalities.

Gate level design methodology currently used for ASIC development has to shift one step over to reach a state of block level design methodology, focusing on the validation of the whole system design.

DOCUMENT CHANGE LOG

Issue/ Revision	Date	Modification Nb	Modified pages	Observations
0/0	26/09/03			First Issue
0/1	24/03/04			comments from ESA integrated.

PAGE ISSUE RECORD

Issue of this document comprises the following pages at the issue shown

Page	Issue/ Rev.	Page	Issue/ Rev.	Page	Issue/ Rev.	Page	Issue/ Rev.	Page	Issue/ Rev.	Page	Issue/ Rev.
all	**0/0**										

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 SCOPE AND OUTLINE OF DOCUMENT 1
1.2 CONTEXT AND RELEVANCE OF THE STUDY 1
1.3 OVERVIEW OF THE STUDY 2

2 APPLICABLE AND REFERENCE DOCUMENTS 3

2.1 APPLICABLE DOCUMENTS 3
2.2 REFERENCE DOCUMENTS 3
2.3 LIST OF ABBREVIATIONS 4

3 DESIGN OBJECTIVES AND CONSTRAINTS 6

3.1 BACKGROUND 6
3.2 DESIGN OBJECTIVES 6
3.3 STUDY LOGIC 7

4 SCOC ARCHITECTURE AND PERFORMANCES 11

4.1 INTRODUCTION 11
4.2 INTERNAL BUS PERFORMANCES 12
4.2.1 System Requirements 12
4.2.2 Bus architecture and performances 14
4.3 DATA TRANSFER 17
4.4 SCOC CONTROL 19
4.5 TIME MANAGEMENT 21
4.6 ARCHITECTURE EVOLUTION 23

5 DISCUSSION ON THE IP CORES 25

5.1 IP PROVIDED BY ESA 25
5.1.1 CTM : CCSDS Time Management 25
5.1.2 LEON : CPU core 27
5.1.3 PTME : Packet Telemetry Encoder 29
5.2 IP PROVIDED BY VENDORS 31
5.2.1 PCI bridge 31
5.3 IP FROM THE ASTRIUM ASIC BACKGROUND 33
5.3.1 PTCDIP : Packet Telecommand Decoder 33
5.3.2 IP1553 : Mil-Std-1553 controller 35
5.4 NEWLY DEVELOPED IP 37
5.4.1 HKPF House Keeping "Packetizer" function 37
5.4.2 AMBA small IP 39

5.4.3 Spacewire *41*

6 ASSESSMENT OF THE DESIGN METHODOLOGY 44

6.1 INTRODUCTION 44
6.2 INITIAL ANALYSIS 45
6.3 ARCHITECTURAL DESIGN 47
6.4 DEVELOPMENT OF AN IP CORE 49
6.5 GATE LEVEL DESIGN 50
6.6 HARDWARE EVALUATION 51

7 ASSESSMENT OF THE USED TOOLS 54

7.1 INTRODUCTION 54
7.2 SIMULATION TOOL 55
7.3 SYNTHESIS TOOL 57

8 EVOLUTION OF STUDY 58

9 CONCLUSION 59

LIST OF FIGURES

Figure 1 : Typical environment of a SCoC integrated in a Spacecraft On Board Controller 13
Figure 2 : SCoC internal bus structure 15
Figure 3 : Performances on CPU AHB bus 15
Figure 4 : Example of Spacewire intelligent DMA 18
Figure 5 : Possibility of DATA transfer with SCoC 18
Figure 6 : Control of the APB bus 20
Figure 7 : Switch Matrix for internal/external synchronisation 22
Figure 8 : Example of local time synchronisation. 22
Figure 9 : Bus evolution 1 – a third AHB bus with associated external memory 24
Figure 10 : Bus evolution 2 – use of Multi-layer AHB 24
Figure 11 : CTM block diagram 26
Figure 12 : A3M use of CTM for time management 26
Figure 13 : Internal structure of LEON1 (extracted from RD1) 28

Figure 14 : Possible evolution of LEON structure....28
Figure 15 : PTME block diagram (example from RD3 with eight input interfaces)....30
Figure 16 : AMBA to PCI interface (from RD5)....31
Figure 17 : Performance on the PCI bus for exchange between 2 BLADE boards....32
Figure 18 : Modifications from PTCD to PTCDIP....34
Figure 19 : Block diagram of the IP1553....36
Figure 20 : IP1553 environment....36
Figure 21 : HKPF block diagram....38
Figure 22 : Possible evolution of the HKPF function....38
Figure 23 : HDMA block diagram....40
Figure 24 : IO Memory Controller Block Diagram....40
Figure 25 : Spacewire IP core structure....42
Figure 26 : Comparison of measured and calculated Spacewire performances....42
Figure 27 : Spacewire information sheet....46
Figure 28 : Overview of the SCoC testbench structure....48
Figure 29 : BLADE board....52
Figure 30 : Implementation of TM and TC emulators in the TEST FPGA....52
Figure 31 : Overview of the Modelsim simulation....56

1 INTRODUCTION

1.1 SCOPE AND OUTLINE OF DOCUMENT

This document is the final report of the contract 13345/98/NL/FM called "Building Blocks for System on a Chip".

Chapter 3 of the document reviews the study objectives and constraints.

Chapter 4 discusses the developed architecture, the choices made, limitations identified and possible evolutions of the architecture. As processor implementation has been limited to LEON-1, the implementation of LEON-2 version is also discussed.

Chapter 5 is a discussion on the IP cores used or developed in the frame of this activity. Difficulties to develop, use or evaluate each IP are assessed. Recommendations for the development and the maintenance of IP cores are also given.

Chapter 6 assesses the methodology 'per work package'. The tools used are also assessed and discussed in chapter 7.

The last part of the final report, chapter 8, proposes evolution based on the experience gained in the frame of this contract, and another contract (A3M), which has used the demonstration board.

1.2 CONTEXT AND RELEVANCE OF THE STUDY

The Frame Contract « Building Blocks for System on a Chip » has been established with ESTEC Microelectronics section to anticipate the necessary evolution of the density and complexity of microelectronics designs.

There are three Call off Orders (COO) within the SoC frame contract. This document reports the work performed for COO3: "Development of a Spacecraft Controller on a Chip". It is based on requirements and functionalities specified in COO2 (refer to AD4).

The selection of the functionalities integrated into the SCoC was based on :

- the generic/standard aspect of the function in space data handling systems,
- the interest of the function.

The objective of the Call of Order 3 (COO3) study has been then established to :

- define a complex SCoC functional architecture
- develop a complex SoC considering the use of existing IP cores and a new methodology for design and integration,
- develop a hardware demonstrator of the SCoC to provide a platform for integration and performances evaluations.

1.3 OVERVIEW OF THE STUDY

The study is conducted as a standard ASIC development, but each phase of the development is assessed regarding its effectiveness for the development of a SoC.

In parallel to the development of the SCoC, a complex IP core - Spacewire interface – was developed.

The COO3 study has been divided into 6 work packages: 1 management (WP3000) and 5 technical (WP3100 to WP3300, WP3600 to WP3700). The work packages 3100 to 3300 correspond to the first phases of an ASIC development: initial analysis, architectural design and detailed design. Work package 3600 corresponds to a hardware evaluation of the design. Work Package 3700 corresponds to the development of the Spacewire IP core. Work packages 3400 and 3500, corresponding to the ASIC backend activities, are not addressed in COO3.

In WP3100, the results of the COO2 activity were assessed and the requirement specification updated according to available functionalities. A first estimation of the complexity of the design led to conclude that the whole functionality would not fit into the selected FPGA. As the purpose of the activity was not to define a methodology for splitting a design into multiple FPGA, it has then be decided to develop an easily configurable SoC, with at least 2 configurations, both of them fitting into a single FPGA (Virtex-E 2000, the biggest commercial FPGA reliably available at the start of the activity).

During WP3200, architectural design, the architecture of the SCoC has been established, leading to the specification of the IP core interfaces. Then the corresponding IP cores have been developed or adapted with different approaches, depending on their source. The architectural phase has been conducted with a progressive integration approach, validating at the same time the possibility to easily configure the SoC. At the end of this work package, a complete RTL database of the SCoC was available and simulated. Re-use of all aspects of a design, and particularly reuse of testbench, has been evaluated and tested during this activity.

WP3300 established the gate level design methodology necessary to handle configurable designs and facilitate the reuse of IP cores.

In WP3600, an FPGA demonstration board was developed, which allows to evaluate the system with IO interfaces specifically used in space applications available on the board : Spacewire, Packetwire, 1553 … The SCoC was then progressively implemented into the FPGA.

It is noticeable that although the design phases are presented sequentially in this introduction, the work has been done in parallel, with a progressive integration of the functionalities into the SCoC.

The definition of the tasks for WP3700, corresponding to the development of a Spacewire interface, is based on two requirements : to provide a full VHDL Spacewire IP core to the European space community, and to evaluate the development needs for a re-usable IP core, to be implemented into the SCoC, but also to be available for other applications. Evaluation of the development needs is performed in terms of documentation, validation procedures, configuration and maintenance.

2 APPLICABLE AND REFERENCE DOCUMENTS

2.1 APPLICABLE DOCUMENTS

AD1 *Call of Order Number 3, ESTEC Contract*, 13345/98/NL/FM

AD2 *Call of Order 3 Statement of Work*, D/TOS/SH/140 issue 2

AD3 *ASIC Design and Manufacturing Requirements*, WDN/PS/700 Issue 2

AD4 *SCoC Requirement specification*, R&D-RP-SOC-214-MMV Issue 3A

AD5 *Spacewire – Links, Nodes, Routers and Networks*, ECSS-E-50-12 Draft 1

2.2 REFERENCE DOCUMENTS

RD1 *The LEON Processor User's Manual*, version 2.4.0, November 2001

RD2 *CCSDS Unsegmented Code (CUC) and CCSDS Time Manager (CTM) Synthesisable VHDL core Data Sheet*, D/TOS-ESM/SH/154 Issue 0.1 Rev. A

RD3 *Packet Telemetry Encoder (PTME) VHDL Model*, PTME-001-01 Version 0.4

RD4 *Spacecraft-Controller-on-a-Chip adapted Packet Telemetry Encoder VHDL Model (SCoC_PTME)*, PTME-002-01 Version 0.4

RD5 *PCI to AMBA Bridge VHDL Model Datasheet*, D/TOS-ESM/153, Issue 2/1, 18 July 2002

RD6 *SCoC Feasibility Study* , R&D -SOC-NT-309-V-ASTR , Issue 1 . 0

RD7 *SCoC Development Plan* , R&D-SOC-NT-303-V-ASTR , Issue 0 . 0

RD8 *SCoC Functional Specification* , R&D-SOC-ST-302-V-ASTR , Issue 0 . 1

RD9 *SCoC Block Diagram* , February 5th 2003 , Issue 8 .

RD10 *SCoC Architectural Design Document* , R&D-SOC-RA-234-V-ASTR , Issue 1 . 2

RD11 *SCoC Simulation Plan and Results* , R&D-SOC-NT-322-V-ASTR , Issue 1 . 0

RD12 *Spacewire Specification and Architecture* , R&D-SOC-NT-292-V-ASTR , Issue 0 . 2

RD13 *Spacewire HW User's Manual* , R&D-SOC-NT-295-V-ASTR , Issue 0 . 1

RD14 *PTCD IP Specification and Architecture* , R&D-SOC-RP-304-V-ASTR , Issue 0 . 0

RD15 *PTCD IP HW User's Manual* , R&D-SOC-RP-299-V-ASTR , Issue 0 . 0

RD16 *HDMA Specification and Architecture* , R&D-SOC-RP-313-V-ASTR , Issue 0 . 0

RD17 *IOMCTRL Specification and Architecture* , R&D-SOC-RP-312-V-ASTR , Issue 0 . 0

RD18 *HouseKeeping Function VHDL Core Description* , R&D-SOC-NT-291-V-ASTR , Issue 0 . 0

RD19 *SCoC Gate Level Design Report* , R&D.SOC.NT.00337.V.ASTR , Issue 0 . 0

RD20 *BLADE Requirement Specification* , R&D-A3M-ST-239-V-ASTR , Issue 1 . 0

RD21 *BLADE User's Manual* , R&D-SOC-NT-314-V-ASTR , Issue 0 . 0

RD22 *BLADE_TEST FPGA User's Manual* , R&D-SCOc-RP-285-V-ASTR , Issue 0 . 0

RD23 *SCoC Hardware Evaluation Plan and Report* , R&D-SOC-NT-327-V-ASTR , Issue 0 . 0

2.3 LIST OF ABBREVIATIONS

AD	Applicable Document
AHB	Advanced High-Performance Bus
APB	Advanced Peripheral Bus
ASIC	Application Specific Integrated Circuit
ASIM	Application Specific Integrated Microsytem
ASSP	Application Specific Standard Product
BC/BM/RT	Bus Controller/Bus Monitor/Remote Terminal (for 1553 component)
BLADE	Board for Leon and Avionics DEmonstration
CCSDS	Consultative Committee for Space Data System
COO	Call Of Order
CPDU	Command Pulse Distribution Unit
CPU	Central Processor Unit
CTM	CCSDS Time Manager
CUC	CCSDS Uniformed Code
DFF	D-Type Flip Flop
DMA	Direct Memory Access
DMS	Data Management System
DSU	Debug Support Unit
EADS	European Aeronotic Defence and Space company
EDAC	Error Detection And Correction
EEPROM	Electrically Erasable Programmable Read Only Memory
ERC32	32-bit Embedded Real-time Computing core
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
FDIR	Failure Detection, Isolation and Recovery
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit, for ERC32
GIC	General Interrupt Controller
GPS	Global Positionning System
HDMA	AHB Direct Memory Access
HK	HouseKeeping
HKPF	HouseKeeping Packet Formater
HSIO	High Speed Input Output
HW	HardWare
I/O	Input/Output
ICD	Interface Control Document
IEEE	Institute of Electrical and Electronics Engineers

IOMCTRL	Input/Output Memory ConTRoLler
IP	Intellectual Property
IP1553	Intellectual Property Block for Mil-Std-1553 bus
IU	Integer Unit, for Sparc
JTAG	Joint Test Action Group (refer to IEEE std 1149.1)
LSIO	Low Speed Input Output
PC	Personnal Computer
PCI	Peripheral Component Interface
PLD	Programmable Logic Device
PLL	Phase Locked Loop.
PPS	Pulse Per Second
PROM	Programmable Read Only Memory
PSS	Procedures, Standards and Specifications
PTCD	Packet TeleCommand Decoder (refers to the MITEL ASIC)
PTCDIP	Packet TeleCommand Decoder Intellectual Property
PTD	Packet Telecommand Decoder
PTME	Packet TeleMetry Encoder
RAM	Random Access Memory
RD	Reference Document
RTL	Register Transfer Logic
SCC	Space Components Co-ordination group
SCoC	Spacecraft Controller on a Chip
SDRAM	Synchronous Dynamic Random Access Memory
SEL	Single Event Latch up
SEU	Single Event Upset
SOC	System-On-a-Chip
SRAM	Static Random Access Memory
SW	SoftWare
TC	TeleCommand
TM	TeleMetry
UART	Universal Asynchronous Receiver Transmitter
VCA	Virtual Channel Assembler
VCM	Virtual Channel Multiplexer
VHDL	VHSIC Hardware Description Language, IEEE standard 1076
VLSI	Very Large Scale Integration (component)

3 DESIGN OBJECTIVES AND CONSTRAINTS

3.1 BACKGROUND

The telecommunication satellite constellation market and the small satellite market drive to dramatically decrease spacecraft equipment weight, consumption budget and recurrent price. The emerging micro satellites and nano satellites generation generates also a demand for much more integrated equipment. ESA and industrials developed in the years 80 and 90 sets of ASICs to reduce the size of electronics. Functions such as the VCA, VCM and PTD used to build a CCSDS TM/TC system, bus interfaces such as MACS, OBDH, 1553, or error correction chips (EDAC, Reed-Solomon, Viterbi…) were manufactured in separate ASICs, and a complete TM/TC system required up to ~10 ASICs.

In the past years, the ASICs developed under ESA contracts were offered to European industrials as ASSP. Most of these chips were designed by using the VHDL language creating software macros that can be reused under certain conditions. The idea by now is to merge all these available blocks in a single ASIC called “System-on-a-Chip” that would be able to perform a large number of the Data Management System functions of a platform.

The System-On-a-Chip is the beginning of a new approach for Space systems, leading to the ASIM (Application Specific Integrated Microsytem) emerging concept. The ASIM will also contain:

- micro mechanical (micro-motor, micro intelligent captor, etc…)
- micro optical (e.g. micro-camera 1024 x 1024 Pixels CCD for Space observation)

3.2 DESIGN OBJECTIVES

The system-on-a-chip approach is a necessary innovation for the European industry. It is obvious that the number of gates per ASIC will continue to grow in the future as it can be seen for commercial applications. There are at least two main domains of applications that require ASICs in Space, which are:

- The digital signal processing for which the available technologies are far from the requirement in terms of gate and in terms of limitation of the power dissipation. These functions are by now far from the System-On-a-Chip concept since they integrate for some of them hundreds of ASICs.
- The Data Management Systems (DMS) composed of elementary ASICs designed by the industrials or bought as ASSP after their development under an ESA contract. These functions can be integrated together with the microprocessor, and this is the next step that is faced the European Space industry to reduce power and mass budget.

The present study focuses on the second domain, the DMS. By now DMS are built around a microprocessor, backplane bus interface, external serial bus interfaces, positioning functions such as GPS,

TM/TC functions... It would be profitable to integrate most of these functions on a single die. The objective is to integrate most of the peripheral ASICs and the controller in a single chip, keeping in mind the Failure Detection, Isolation and Recovery (FDIR) scheme in the implementation.

The developed System-On-a-Chip can be used in most of the satellite platforms, and especially it would be convenient to use for small satellites.

The System-On-a-Chip developed in this activity is a one chip Spacecraft Control System FPGA comprising the following functions:

- LEON SPARC CPU with FPU
- Backplane bus controller (PCI)
- Spacewire Bus Interfaces
- CCSDS packet telecommand and telemetry system
- TM Housekeeping generation
- 1553 System bus controller/bus monitor/remote terminal
- Time management system

The subject of this study is at first to validate the concept of a System-On-a-Chip approach. The work to carry out in this study is to practically try out the problems posed by the System-On-a-Chip approach for a Space application. It follows as a basis the classical development plan of ASIC that is defined by ESA. Since an ASIC foundry of such size is very expensive, a first step using large FPGA for prototyping of the System-On-a-Chip is made in this phase of the contract. The System-On-a-Chip is implemented on a demonstration board, and the ASIC is integrated in a XILINX VIRTEX-E FPGA.

3.3 STUDY LOGIC

The logic of the study is described hereafter. In the previous phase of the contract called COO2, the Design Initiation was performed. A requirement specification of the System-On-a-Chip was issued.

COO3 Contract starts with the Initial Analysis that is divided in two parts:

- A study of the System-On-a-Chip to be developed in terms of functions to be implemented, in terms of communication between functions (throughput of the busses, gate count, pin count).
- An analysis of the problems posed by the development of a System-On-a-Chip which are briefly the quality of the used IP blocks, the functional verification of the blocks when inserted in the System-On-a-Chip, the testability, the gate-level design….

A Feasibility Study report has been written (RD6). A functional specification has also been written (RD8). This latter is not a self-contained document due to the complexity of the various functions embedded. The SCoC specification refers to the specifications of the elementary functions that build it. During this task, the specification of the BLADE hardware evaluation board has also been issued (RD20).

Secondly, the Architecture Design has been carried out:

- A study of the backbone bus of the System-On-a-Chip and the connection of all the VHDL modules together is done.
- The IP blocks available from Astrium SAS are from 3 sources:
  - The TC function is issued from the PTCD ASIC. PSS45/SVE protocols and in-test functions are removed, IO buffers are removed and the original testbench is run to check TC core functionality. Minor function changes asked by ESA are made, and the original testbench is run once again. TheTC function is inserted in the APB, APB HK and AHB interfaces. At this level only interfaces are tested since the TC core itself is unchanged. The complete PTCD test suite is not ported to the APB/AHB new format. TC IP block is ready to be inserted in the SCoC at this level.
  - The 1553 BC/RT function is provided ready to be inserted in the SCoC with a test suite and emulators allowing testing it.
  - New blocks, such as Spacewire IP and Housekeeping Packetizer, are developed from a specification, and a testbench allowing exercising their functionality outside SCoC is made.
- The IP blocks coming from ESA are associated with their testbench and their documentation. Astrium receives them; their testbench is run, and their compatibility with the on chip bus checked. After this step they are ready to be inserted in the ASIC.
- The problem of the SEU protection is not addressed on the existing blocks. Some of the blocks provided by ESA include SEU protection, other no. TC function does not include SEU protection, but in case of ASIC made for the TC function, the best solution would be to triplicate all the sensitive D flip-flops. Thus design for SEU will not be made on existing code.
- The necessary glue logic between the blocks is coded in VHDL.
- At this step all the blocks are in the correct form to be included in the System-On-a-Chip. The assembly can be done and functional vectors are run at the top level. Only a partial test suite is run at SCoC level. It is assumed that blocks are validated at their level.
- An Architectural report is written (RD10).

In parallel, the design of the demonstration board is made. The schematic of the board is created, the PCB is designed, and the demonstration board is manufactured and populated.

Thirdly, Detailed Design is performed :

- All the classical operations of the Detailed Design are made: Synthesis, layout and routing, static timing analysis and gate level simulation, with XILINX VIRTEX FPGA as target. Gate simulations are performed on a subset of the vectors. Timing analysis is completed.
- A Gate Level Design Report is written (RD19).

Finally, hardware evaluation phase is carried out:

- The debug of the demonstration board is performed. In this phase, a complete validation is not envisaged. The chip and the board are debugged by running elementary software comparable to the one used for simulation. Each interface is activated, but its full functionality not validated. This step allows validating the concept of implementing the System-On-a-Chip in the PLD, demonstrates the functionality of the internal bus, and verifies the principle of the communication between blocks and ports.
- At the end of this phase, a Hardware Evaluation report is written (RD23).

In practice, the previously described tasks are performed concurrently and the VHDL blocks are progressively integrated in the SCoC. Thus, it is possible to start with a SOC including LEON+CPU bus structure+IO bus structure+1553 and test it on the board. During the test, the next block can be integrated at VHDL level (Spacewire for example) and the SOC hardware is then updated to test the newly integrated block.

This approach needs at first to perform a detailed architecture study in order to dimension the glue logic of the SOC and to assess the performances.

The advantages of such approach are multiple:

- The time of development can be decreased by tailoring the tasks (like on a production line)
- Simulation validation and hardware debug is facilitated (a bug appearing at insertion of a new block is more easily circumvented)
- In a hardware/software approach, applying the same logic to the software development can decrease the time of development by validating the software in advance (without waiting for the whole hardware integration).

The use of a standardized scalable internal bus structure is obviously a pre-requirement of this approach.

Page intentionally left blank

4 SCOC ARCHITECTURE AND PERFORMANCES

4.1 INTRODUCTION

The architecture of the SCoC is developed with modularity to facilitate the insertion of new blocks or the suppression of others. This modularity is based on the internal bus structure and on standardized services provided to the blocks, like interrupts management and time distribution, independently of their intrinsic functionality.

This section focuses on particular points of the SCoC architecture: internal bus, data transfers, time management. Some propositions of architecture evolution are also made in order to cope with systems more demanding in terms of performance.

The developed SCoC is based on standard requirements of data throughput for data handling systems. The internal bus architecture is based on these requirements. The section 4.2 describes the requirements taken into account, the constraints for the internal bus structure, and presents the implemented architecture along with its performances.

The internal bus provides the pipelines for internal data transfers. The global mechanism activated to handle these transfers impacts on the overall system performances. The section 4.3 provides the rationales of the data transfer mechanisms and gives some examples of implementation.

The section 4.4 then focuses on a different aspect of the overall system architecture: time management and synchronisation. The proposed SCoC embeds basic scalable structures providing services to the IP cores implemented, such as time synchronisation, slot synchronisation, event time tagging.

The SCoC architecture exploration phase was limited for schedule and duration aspects, and also due to the architecture of embedded functions, such as LEON, that were not suited for architecture modification. The section 4.6 provides some possibilities of internal architecture evolution, particularly focused on the internal bus structure.

4.2 INTERNAL BUS PERFORMANCES

4.2.1 System Requirements

The performances of each SoC interface function taken individually can be estimated and controlled. The architecture choices for the internal block implementation have only local repercussion. Definition of the SoC internal bus architecture has a drastic impact on the performance of each block taken individually, and also on the overall performance of the system.

Bus standard

A driver for the definition of the internal bus was its availability on the commercial market as a standard, preferably open and free of access. A lot of standards for on chip bus were existing, all responding to the SoC requirements and the selection of the AMBA™ bus was linked to the development of the LEON processor by the ESA at the start of the activity.

Rationale for architecture

The internal bus is the main pipe for data exchanges between peripherals and CPU. The selection of this bus, performed with respect to criteria such data throughput, latency, structure (multi-master, multi-slave), availability of IP cores compatible with the bus, greatly impacts the performances of the SoC.

The challenge for the SCoC is that the internal bus performance requirements greatly depend on the scope of application of the system and on the actual usage of the interfaces.

For example, using the PCI bus for communication with a thermal control module does not request the same performances requirements as using it for communication with a Mass Memory. The same applies for the Spacewire interfaces, which allow data transfer rates as low as a few bytes per second up to hundreds of megabytes per second.

In addition, having data transiting from the Spacewire to the PCI through the SCoC does not request the same performances and architecture as for data received from the Spacewire and processed by the CPU.

The selection of the bus architecture is also driven by the architecture of data memories. Indeed, the performances of each interface of the SCoC would require an external buffer memory (SRAM for example), but the pin count of the overall system has to stay compatible with the available and qualified ASIC packages. This latter requirement leads to group together interface buffer memories, and then to create a single bus on which interfaces IP core are connected.

Finally, the internal requirements of IP cores, such as maximum latency, put constraints on the busses that also drive the overall architecture.

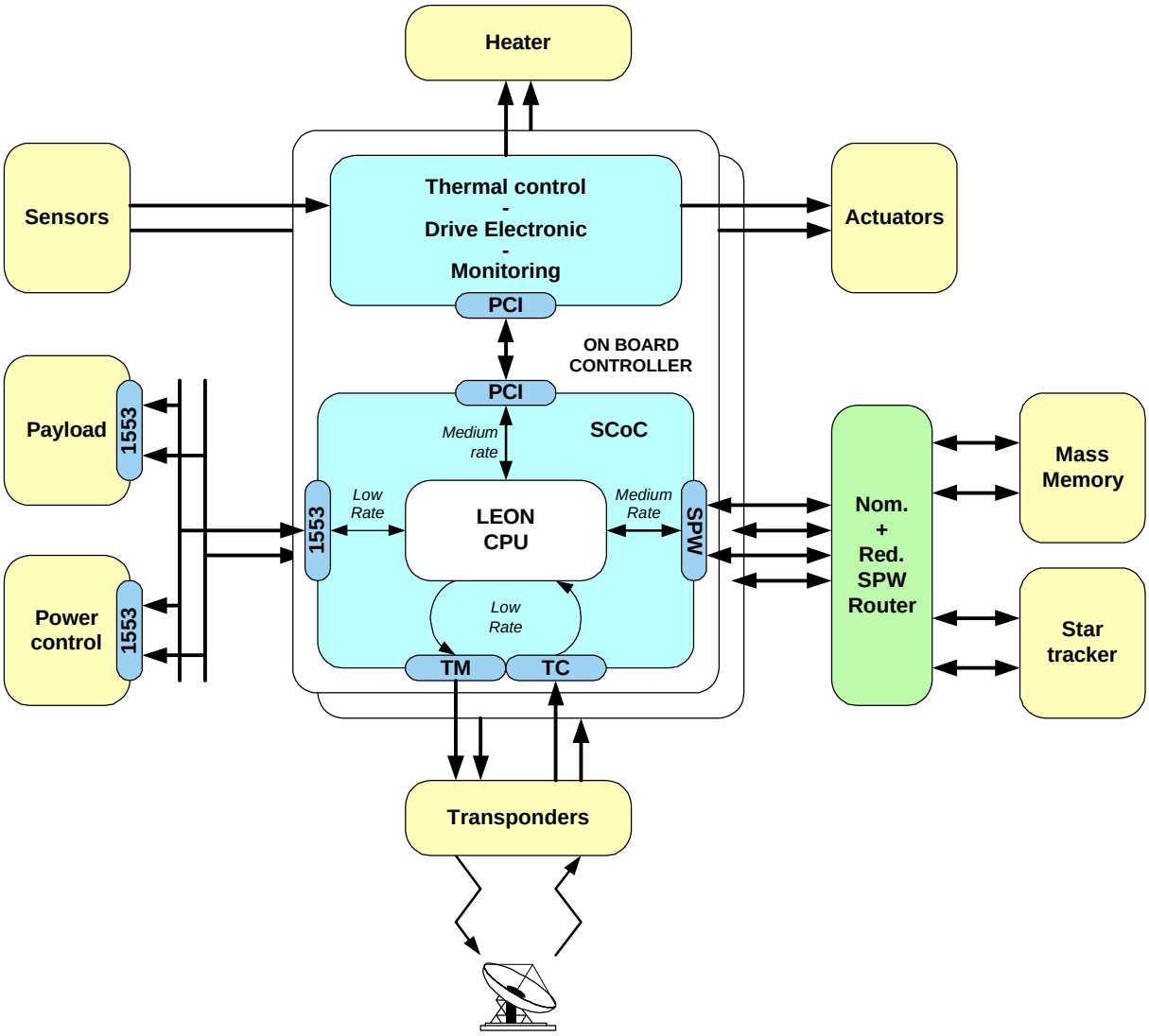

Figure 1 : Typical environment of a SCoC integrated in a Spacecraft On Board Controller

Knowledge of the system requirements is necessary for the construction of the SCoC bus architecture. The performance requirements at system level deeply impact the SCoC architecture. In this example of SCoC usage in a typical On Board Controller, the transfer rates at each of the SCoC interfaces are low and medium. All data are handled by the Central Processor Unit (LEON CPU)

4.2.2 Bus architecture and performances

The analysis of a standard system requirement drives the SCoC bus architecture selection and leads to a choice of 2 AMBA AHB busses. The performances of this architecture for the targeted application are good. Hardware evaluation platform provided with the BLADE board is an excellent tool for architecture implementation and evaluation.

Architecture selection

The main drivers of the architecture selection are:

- The SCoC defined for this study is the Spacecraft Controller. Most of the data that pass through the interfaces come from or are destined to the CPU (LEON). For this, all data shall pass through a storage area (data pool) directly accessible by the CPU. This storage area is external to the SCoC in order to cope with large capacity.
- 2 types of interfaces, Spacewire and PCI, provide high data throughput to the CPU, and have flow control on their external bus that allow handling of high bus latency without the need of large FIFOs,
- The other interfaces, 1553, TM and TC generate low data throughput with the CPU, but do not have flow control and then need an accurate latency control for access to their buffer memory.

These considerations led to the specification of an architecture with 2 internal busses, a DMA bridge in between and two external memory controllers, as depicted in Figure 2 :

- The CPU bus connects together the LEON CPU, the PCI and the Spacewire blocks to the CPU memory. This bus is arbitrated with fixed priorities with the CPU having highest priority.
- The IO bus connects together the 1553, the PTCD and the PTME to an IO memory dedicated to these functions. This bus is arbitrated with round robin priority guaranteeing access latency for each function, as each block has a defined maximum transaction length.

Architecture performance

The selected bus architecture leads to a good performance on the CPU bus for application of SCoC in a data handling system. However, it is not possible to guarantee a high CPU performance while having high data throughput on the PCI or Spacewire interfaces. PCI and Spacewire are independent (DMA) masters on the bus, but performance has to be shared between CPU computation and interface transfers. As the selected SCoC is destined to Spacecraft Control, the architecture privileges the CPU computation, by putting the CPU on the highest priority level on the bus. No hardware mechanisms are provided to accurately share the performance between the CPU and Data transfer rate. The possibility to have high data transfer rate on this system relies on the cache performances of the CPU and its associated software. The theoretical performance of the AHB bus is assessed by the means of a spreadsheet calculation, and Figure 3 shows a typical trade-off between CPU and interface performance.

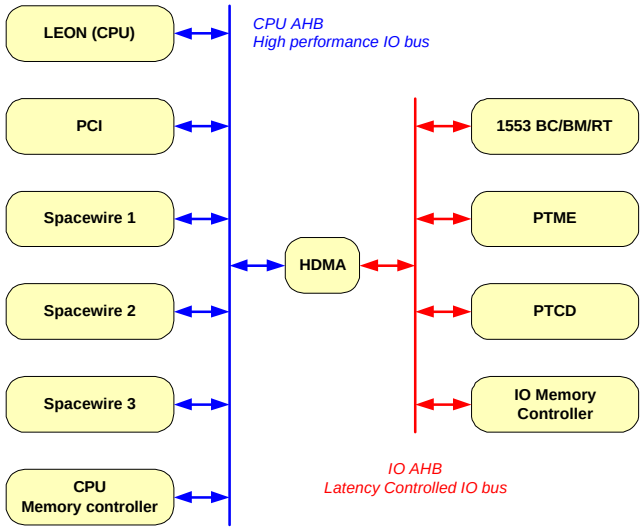

Figure 2 : SCoC internal bus structure

Two AHB busses, the CPU AHB for high performance, flow controlled interfaces, and the IO AHB for medium data rate interfaces, form the basics of the SCoC internal bus structure. LEON CPU is the highest priority master on the CPU bus, and then the available bandwidth for PCI and Spacewire shall be allocated by software control of the CPU bus activity.

COMPUTATION OF PERFORMANCES ON CPU AHB BUS

Global Hypothesis	
Maximum allowed bus load	80%
Bus Frequency	100 MHz

Slaves Hypotheses	
Memory controller	
RAM Read Wait states	0
RAM Write Wait States	0
MCTRL AHB WS on first read access	3
MCTRL AHB WS on next read access	2
MCTRL AHB WS on first write access	3
MCTRL AHB WS on next write access	2

Masters Hypotheses	
Processor	
CPU usage ratio	**65.5%**
Instruction cache hit ratio	80%
Data cache hit ratio	80%
Load instruction ratio	10%
Store instruction ratio	5%
Instruction cache fill burst length	4
Spacewire	
overall Spacewire TX bit rate	100 Mbits/s
overall Spacewire RX bit rate	100 Mbits/s
PCI	
PCI write rate	0.5 Mwords/s
PCI read rate	0.5 Mwords/s

Data Path	**Dir**	**Burst len**	**clk period**	**Burst freq**	**BW allocated**	**Ratio**
Instruction cache fill	R	4	9	3.273	29.458	29.5%
Processor data load	R	1	3	6.546	19.639	19.6%
Processor data store	W	1	3	3.273	9.819	9.8%
Spacewire transmission	R	1	3	3.125	9.375	9.4%
Spacewire reception	W	1	3	3.125	9.375	9.4%
HDMA from CPU to IO	R	4	9	0.004	0.035	0.0%
HDMA from IO to CPU	W	4	9	0.005	0.049	0.0%
PCI reads	R	4	9	0.125	1.125	1.1%
PCI writes	W	4	9	0.125	1.125	1.1%
Total					**80**	**80.0%**
Requested					**80**	**80.0%**
Slack					**0**	**0.0%**

Figure 3 : Performances on CPU AHB bus

Considering a system working at 100 MHz, and a software cache utilisation as good as 80% of cache hit, it is necessary to limit the CPU usage ration to 65% in order to guarantee an average 100 Mbits/s full duplex transfer rate on the Spacewires and 0,5 Mwords/s full duplex on the PCI

4.3 DATA TRANSFER

The integration of complex IO interfaces within the same chip as the CPU needs a study of data exchanges mechanisms between the peripherals. In the SCoC, DMA mechanisms have been spread over each function. This allows having high and efficient transfer rates with a little use of the CPU.

Intelligent DMA are integrated to the function.

The DMA functions can be either concentrated in a function with multiple channel of generic DMA, or can be integrated in the IO peripheral IP and dedicated to specific needs of the IP. The latter has been implemented in SCoC. The architecture of SCoC integrates a lot of DMA transfer functions (see Figure 5):

- Spacewire (see Figure 4):
  - The transmitter host interface integrates a specific DMA transmitting linked list of packets (with a C-like memory structure). Size of the linked list is limited only by the available memory.
  - The receiver host interface integrates a DMA function with dynamic flip-flop buffer management. Buffers can be mapped anywhere on the AHB address range, having arbitrary size.
- PCI:
  - The PCI bridge embeds a DMA function on both, the initiator side (controlled by the ScoC CPU) and the target side (transfers initiated by other masters on the PCI bus) allowing to handle any transfers between the PCI bus and the AHB bus.
- Mil-Std-1553 BC/RT :
  - The Bus Controller directly accesses to the IO memory to execute a program and to access data structure in the memory. Program is standalone and can execute in parallel of the CPU activity.
  - The remote terminal handles the I/O memory and stores and retrieves data autonomously, based on characterization information stored into the same memory. It is possible to allocate reception and transmission buffer for each sub-address of the bus.
- HDMA :
  - The HDMA (A**H**B **D**irect **M**emory **A**ccess) acts as a bridge between the CPU and the IO AHB busses. Being a master on both busses, programmed by the CPU, it permits the transfer of data structure between the two busses. As the transfer rates on the IO bus are low (and then the associated amount of data to process is low), a simple single channel DMA function has been implemented.

This variety of DMA function (see Figure 5) allows transferring efficiently data from/to the CPU memory or between peripheral I/Os. For example, it is possible to use the PCI DMA or the Spacewire DMA to directly transfer data from a PCI agent to the Spacewire link. Data received from the Spacewire can also be directly written to a PCI agent. Not all of the possible data transfers are really meaningful, as shown in the yellow/orange fields in Figure 5.

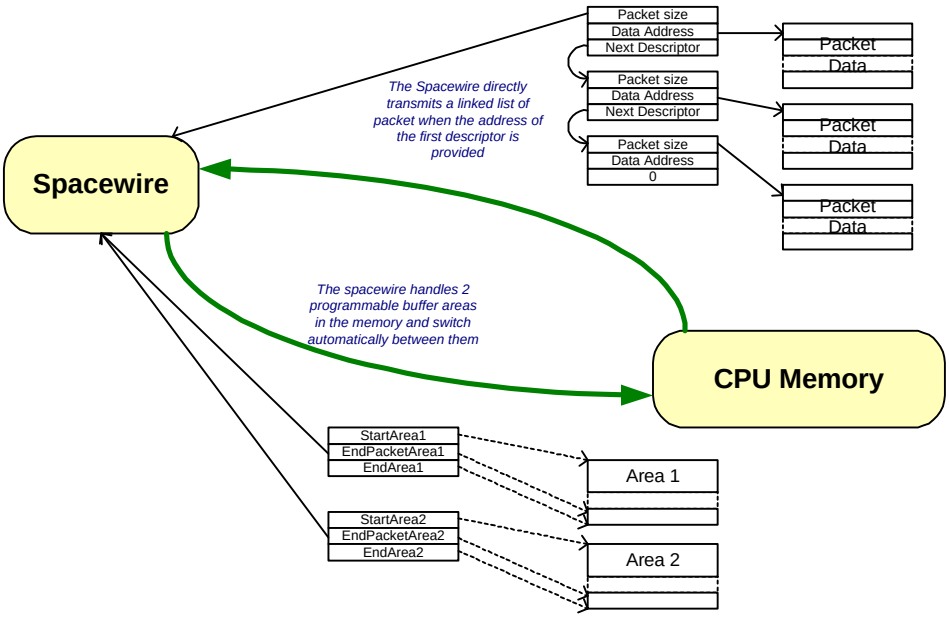

Figure 4 : Example of Spacewire intelligent DMA

The linked list of packet for transmission can be directly generated as a standard C-like linked list. The double area with flip-flop management in reception allows using the Spacewire at full performances

from \ to	Spacewire	PCI	Config. (APB)	CPU memory	TM	IO memory
LEON	write	write / read	write / read	write / read	write VC1 / VC2	-
Spacewire	-	write / read (SDMA)	possible but not recommended	write / read (SDMA)	possible but not recommended	-
PCI	write direct or PDMA	-	write / read	write / read direct or PDMA	write VC1 / VC2 direct or PDMA	-
CPU memory	write / read (SDMA)	write / read (PDMA)	-	-	-	write / read (HDMA)
1553	-	-	-	-	-	write / read
TM	-	-	-	-	-	write / read
TC	-	-	-	-	-	write / read
IO memory	write (HDMA) (low interest)	write / read (HDMA)	possible but not recommended	write / read (HDMA)	write VC1 / VC2 HDMA (low interest)	-

Figure 5 : Possibility of DATA transfer with SCoC

The DMA and bus structure of SCoC allow transferring data within SCoC and with external peripherals in various ways. For a given transfer direction it is sometimes possible to use different strategies, thus adapting the needs to the system requirement. HDMA stands for 'use of HDMA function', SDMA for 'Spacewire DMA function' and PDMA for 'PCI DMA' function.

4.4 SCOC CONTROL

The SCoC is mainly controlled through the APB bus. All IP cores are connected to this bus for register acces. This structure allows the control of the SCoC by any AHB master connected to the CPU AHB bus. Basically the LEON CPU core acts as the main system controller, but the SCoC can also be configured as a PCI agent and be controlled the the PCI system controller (or any master agent).

Architecture of control:

All IP cores, whatever they are connected to the CPU AHB or IO AHB bus, are connected to the APB bus for access to their internal registers. The APB Master block acts as the master of this bus, and is connected as a Slave to the CPU AHB bus. Then any master of the CPU AHB bus can initiate exchange on the APB memory area. This is particularly true for the LEON CPU core and the PCI core.

The structure of the Host interface of the Spacewire block does not allow random access AHB bus, but a modification of the interface is possible thanks to the modular implementation of the Spacewire Block.

Access to the APB bus automatically implies access to the Telemetry virtual channels 1 and 2 implemented in the PTME. Then any CPU AHB master can directly transfer telemetry without CPU support.

Example of SCoC controlled by external PCI agent:

The access to the APB bus throught the PCI is particularly useful either for usage in flight environment or for monitoring of the SCoC. Here are some example of usage of the SCoC internal resources through the PCI.

<u>Spacewire interface :</u>

An intelligent external PCI agent can use one (or more) of the Spacewire interfaces. For this, the agent can prepare the data either in the CPU memory or in a memory accessible by the PCI initiator and program the Spacewire TX DMA through APB accesses.

The PCI agent can also program the memory area used by the Spacewire RX DMA, and map it directly on the PCI address range. The data received from the Spacewire are then directly transmitted through the PCI interface.

<u>TM/TC control</u>

The PTME virtual channels 1 and 2 are accessible through the APB bus. Then a PCI agent can directly insert telemetry packets in the telemetry transfer frame. Abnother solution is to store the telemetry data in the context memory of the HKPF function. That telemetry can then be transferred as house keeping telemetry at the occurrence of any external event.

<u>1553 management</u>

Access to the APB bus allows programming DMA transfers between the CPU AHB bus and the IO AHB bus. It is then possible to use the 1553 controller from an external PCI agent.

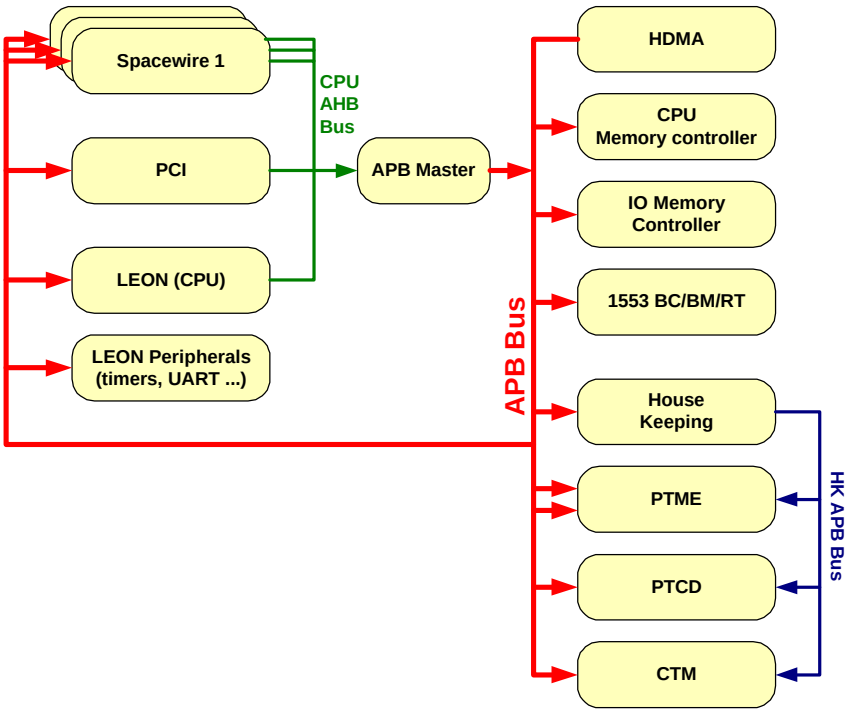

Figure 6 : Control of the APB bus

The APB Bus allows accessing to all registers implemented in the different IP cores. Any master on the CPU AHB Bus can access to the APB bus as as Master through the APB Master interface. It is then possible to control the SCoC through the PCI interface for example.

4.5 TIME MANAGEMENT

The Time management and time distribution is not a specific issue of the SCoC. But the SCoC has to provide a flexible synchronisation between peripherals in its architecture, as the design can be frozen in an ASIC. A synchronisation switch matrix is implemented into the SCoC to adapt the function to specific system requirements.

Time synchronisation is required at different levels in the data handling and AOCS subsystem of a spacecraft. As SCoC embeds a lot of peripheral interfaces, the local time and synchronisation management and the time distribution outside of the SCoC has been addressed.

Local time management

SCoC integrates the ESA CCSDS Time Management IP core (CTM). This core allows handling a copy of spacecraft local time with the CCSDS Unsegmented Code (CUC) standard. The CTM can then be programmed to generate Pulse Per Second (PPS), fixed alarms, repetitive alarms, and to sample local time on occurrence of events. In addition, a Time Packet structure can be inserted automatically (with no software action) in the ground telemetry data flow.

Time synchronisation and distribution

The local time can be correlated with an external central time management system with a limited action of the software. The CTM receives a correlation pulse (PPS for example) associated to a time value message that can be received through any SCoC interface. The CTM can also act as a central time system and the CPU can generate time messages while the CTM automatically generates the PPS. This PPS can then be transmitted directly as a pulse through an external interface, or can be transmitted through a Spacewire time code message or a Mil-Std-1553 Synchronise mode command.

Time Tagging

The CTM offers the possibility to time tag events inside or outside of the SCoC. The internal events can be as various as time code reception from the Spacewire, time code pulse synchronisation from the PTME (synchronisation with the ground system), Synchronise command received from the Mil-Std-1553 bus, or message received from the telecommand (PTCD). Requirements of time tagging can vary from systems to others.

Real Time Cycle management

The CTM alarm generation can be connected to various synchronisation signals inside of the SCoC. For example, the time code generation of the Spacewire, the synchronisation of Mil-Std-1553 Bus Controller exchanges or the automatic generation of house keeping telemetry reports. Requirements for real time cycle management can vary from systems to others.

Switch Matrix procures system flexibility

A programmable switch matrix implemented into the SCoC handles the different possibilities of synchronisation described above. This switch matrix implements possible resynchronisation of asynchronous signals, conditioning and routing to the programmed IP core input. The switch matrix is described in RD10.

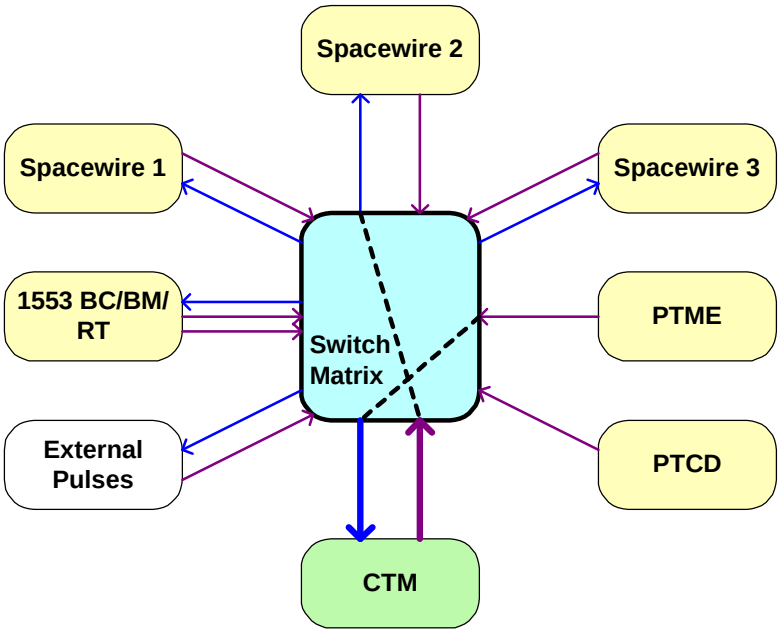

Figure 7 : Switch Matrix for internal / external synchronisation

Integrated IO IP cores within the same chip require adding flexibility for synchronisation and time distribution. The figure indicates how the switch matrix can be used to connect for example the Spacewire time code generation input to the CTM PPS output and the PTME time code output to the CTM event tagging input.

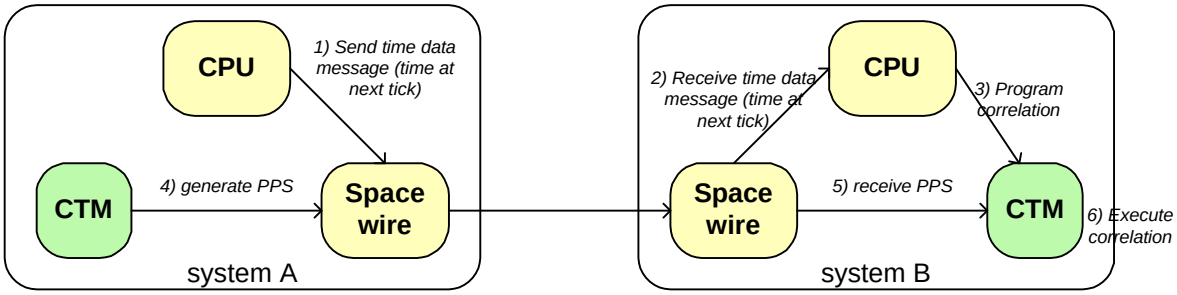

Figure 8 : Example of local time synchronisation.

System A integrates central time management and distributes time to other systems. For a direct Spacewire connection, accuracy of time synchronisation has been measured below 10 us between 2 BLADE boards (reference: A3M study)

4.6 ARCHITECTURE EVOLUTION

The basic architecture of SCoC has been evaluated in terms of performances. Bottlenecks have been identified and are subject of recommendations for a future implementation of a SCoC in hardware. Evolutions mainly consist in an upgrade of LEON CPU and a modification of the internal bus architecture to offer simultaneously higher performances in terms of computation and data transfer.

From LEON1 to LEON2

Facing the continuous evolution of a LEON CPU, it has been decided at the beginning of the SCoC activity to freeze the version of the processor core implemented into the SCoC. The version implemented is the last stable version of LEON1 (version 2.4.0). Modifications in LEON2 provide important performances upgrade. The most important one is the cache management. Indeed, it has been observed during the tests that the cache performances drastically impact the overall system performances, as the CPU bus is shared with other peripheral IP cores. For example, the lack of cache snoop in LEON1 reduces the performances for Spacewire or PCI data management by CPU (one should not allow those peripherals to write into a cacheable area). The Debug Support Unit (DSU) also provides important improvements that ease the debug and verification of the system.

Dedicated bus for High Speed IO

The performance bottleneck of the system is identified at the CPU AHB bus level, providing access to the CPU memory. CPU computation performances, considering a poor cache statistic, deeply rely on this bus performance, as well as the Spacewire and PCI data flow rate. Different architectures can be defined to improve the performances in terms of CPU computation and peripherals transfer rates. These architectures mainly rely on a third bus. The example provided on Figure 10 is expected to exhibit better performances, but the solution would have to be evaluated. The solution can be implemented with existing IP cores.

Use of Multi-Layer AHB interconnect matrix

Following the increasing complexity of SoC designs, ARM has developed a Multi-Layer AHB specification. In this configuration, an interconnection matrix replaces the classical AHB arbiter. The advantage of this solution compared to the 3 busses architecture presented before is that it suppresses the need of AHB bridges between the busses, and the drawbacks associated to bridges such as latency. Performance upgrade is expected to be high with this kind of architecture.

The presented bus evolutions imply the addition of one additional memory controller for external SRAM/SDRAM, and then an increase of the pin count of the SCoC. This might impose difficulties finding a suitable package.

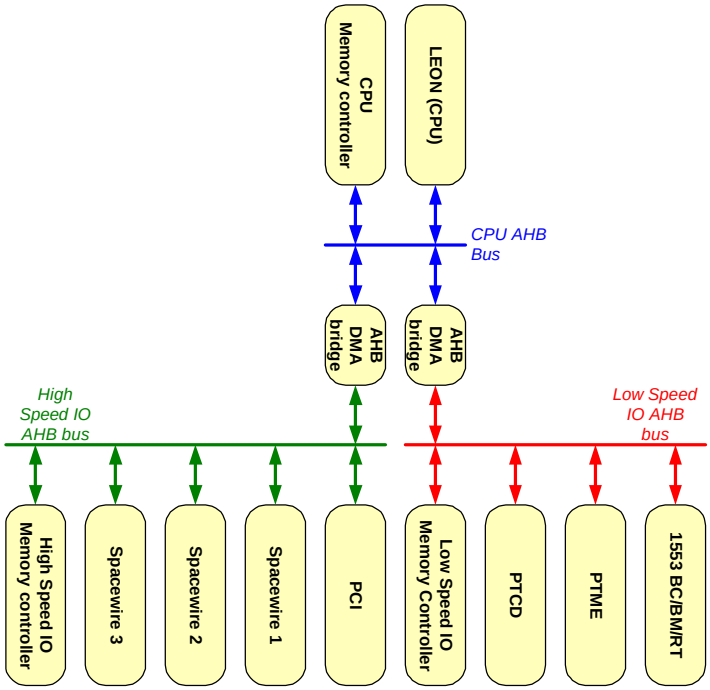

Figure 9 : Bus evolution 1 – a third AHB bus with associated external memory

Adding an AHB bus with associated external memory for high speed peripherals would improve system performances for CPU processing capabilities and transfer data rate on Spacewire and PCI. The counterpart is an increase of external memories and chip pin count.

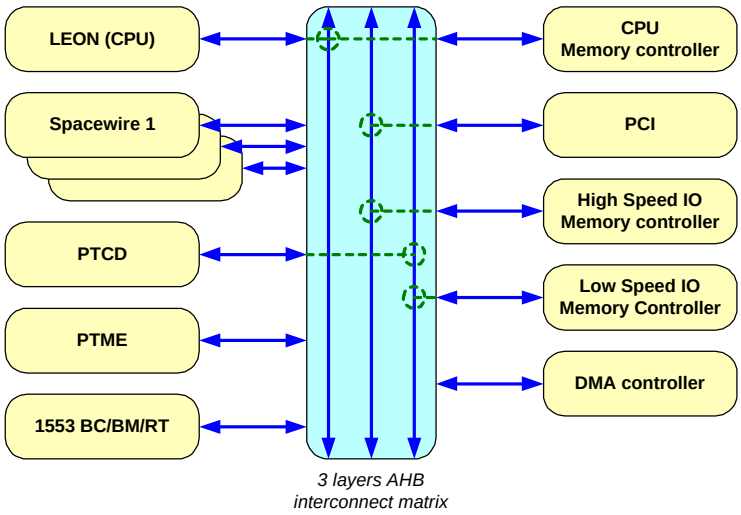

Figure 10 : Bus evolution 2 – use of Multi-layer AHB

The 3-layer AHB arbiter allows at the same time LEON reading instruction to its CPU memory, PCI writing to the High-Speed IO(HSIO) memory and PTCD accessing to the Low-Speed IO (LSIO) memory. At the next cycle, the interconnect matrix can for example connect the CPU to the HSIO memory to retrieve data previously written through the PCI.

5 DISCUSSION ON THE IP CORES

We can distinguish 4 main sources for the IP cores used within the Spacecraft Controller on a Chip : IP provided by ESA, commercial IPs, Astrium ASIC design modified as IP cores, and IP fully developed in the frame of this activity.

5.1 IP PROVIDED BY ESA

5.1.1 CTM : CCSDS Time Management

The CTM provides all the functionalities for the on board CCSDS Time Management by integrating time keeping, alarms and pulse per second generation, event time tagging, time correlation and re-synchronisation. Its use within a complex test structure in A3M study confirmed the effectiveness of this system.

Description of the IP

The CTM block developed by ESA can be used within a System on a Chip for local time keeping. In addition, it can be used as the on-board central time reference provided that an accurate time reference is available. Its 2 level structure (a CCSDS Unsegmented Code core structure surrounded by an APB and services wrapper, see Figure 11) allows adapting the interfaces without affecting the core functionality. The service wrapper integrates an automatic Time Packet generation for the SCoC housekeeping telemetry function.

Available documentation

The CTM documentation consists in a single document “CCSDS Unsegmented Code (CUC) and CCSDS Time Manager (CTM) Synthesis able VHDL core Data Sheet” (RD2) fully explaining the functionality, usage and interfaces of the CTM. In addition, an Excel sheet is provided to compute the programming parameters for the software. This sheet is not documented, but relies on formulas given in RD2.

Success

After basic tests of the CTM functionalities by simulation and on the BLADE hardware platform, the CTM has been used by the A3M project to time tag events within the SCoC. The A3M project used 3 BLADE boards connected to each other through Spacewire link (see Figure 12). The CTM has been used to keep an accurate time information on each node. Synchronization of the 3 nodes was possible with the CTM functionalities of re-synchronization. Final accuracy of time between each node was better than 10 µs.

Problems encountered

Apart from a minor bug on the pulse per second generation, detected during hardware evaluation of the SCoC and easily corrected, there was no problem encountered with the use of the CTM function.

Possible evolutions

A CPU action is required to perform time re-synchronization or correlation. It can be interesting to consider a full hardware integration of the function.

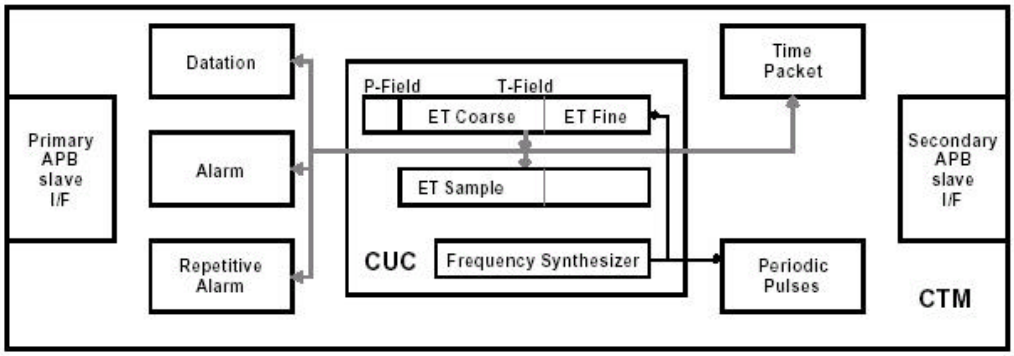

Figure 11 : CTM block diagram

The 2 level structure of the CTM – internal CUC integrating basic functionalities and CTM wrapper implementing APB interfaces and services – eases the adaptation of the CTM to specific applications

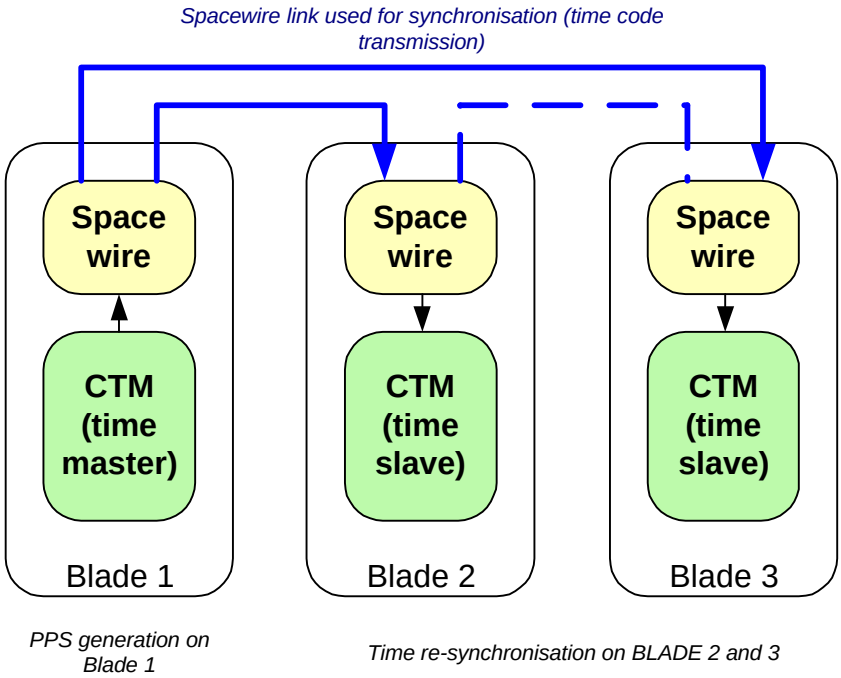

Figure 12 : A3M use of CTM for time management

The functionalities of the CTM allow using it as a central time function distributing time reference or as a local time function re-synchronised to the central time. The CTM core is the same for both functionalities

5.1.2 LEON : CPU core

LEON-1 SPARC V8 core developed by ESA and Gaisler Research provides a powerful highly configurable CPU core available for integration within System on a Chip. Nevertheless, while the configuration is efficient, it is not adapted to the use within a SoC, as LEON core can be seen itself as a SoC.

Description of the IP

LEON IP provides full functionalities of a SPARC V8 processor core with embedded cache for instruction and data, Integer Unit, interface for Floating Point Unit and Co-processor. In addition, lot of peripherals such as UART, timers and watchdog, memory controller are integrated. LEON processor embeds an internal AMBA AHB and APB bus architecture (see Figure 13).

Available documentation

LEON IP core is fully described in a single document : “LEON1 VHDL Model Description” (RD1). In addition, a public newsgroup is open and is a powerful source of information and discussion around the core development and utilization.

Success

The main success with LEON use in SCoC is that the first version of SCoC implemented on the BLADE hardware evaluation platform was fully operational. The development suite provided with the IP allows a rapid development of basic software test.

Problems encountered

The main drawback of LEON VHDL structure comes from the fact that LEON is already a System on a Chip and that the configuration of LEON functionalities (cache, IU, FPU) is merged with the configuration of the SoC structure (AHB and APB structure, small peripherals such as memory controller and UARTs). Then it is not possible to implement Rather than instantiating LEON as an IP in a SoC, this means that the SCoC components have to be implemented as additional peripherals into the LEON structure. This situation led to freeze the version of LEON used at the beginning of the project : LEON1 2.4.0.

By freezing the version of LEON, we were not able to benefit of the evolution of CPU/cache performances provided with LEON2. The problem of cache snooping, software investigation with the Debug Support Unit, are all resolved with LEON2.

Possible evolutions

In addition of an evolution of SCoC to integrate LEON2, we need to envisage a complete re-formatting of LEON configuration mechanism in order to explicitly separate CPU core configuration from SoC configuration (bus structure for example). See Figure 14.

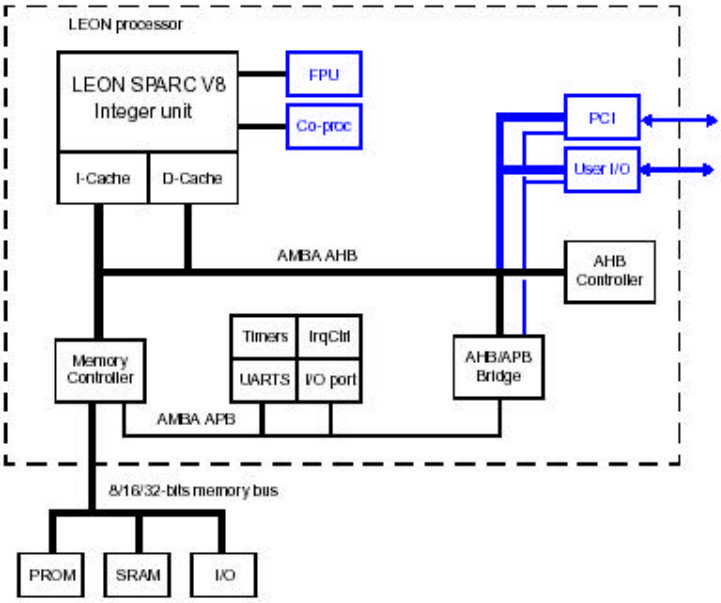

Figure 13 : Internal structure of LEON1 (extracted from RD1)

LEON processor embeds all functionalities of a CPU core. The main difficulty is that LEON can be considered as a SoC and is hardly integrated within another SoC design such as SCoC.

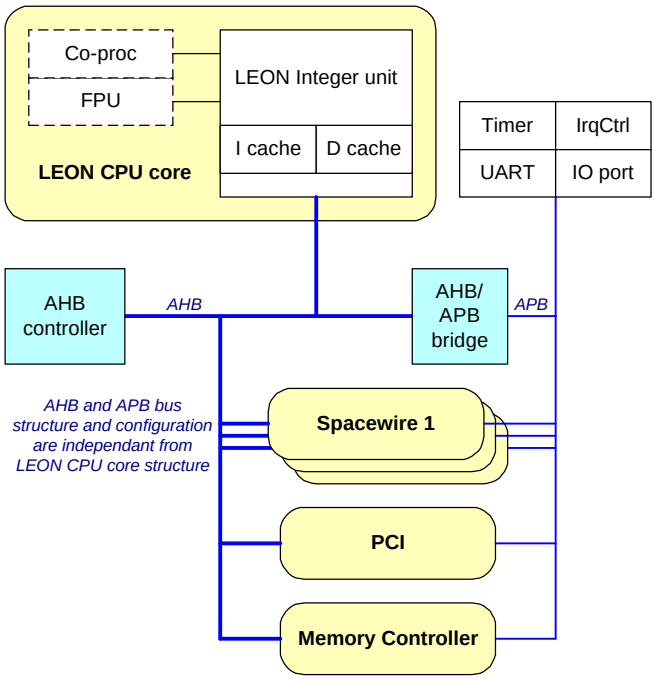

Figure 14 : Possible evolution of LEON structure

Separating the configuration of AMBA bus structure from the LEON CPU core configuration eases the development of the SoC structure, as well as the possibilities for architecture exploration and evolution. Updates of the LEON CPU core can be integrated more easily.

5.1.3 PTME : Packet Telemetry Encoder

The PTME provides all the functionalities of the ground telemetry system, integrating the previous design of ASSP VCA, VCM, Rescue and Turbo Encoder. The functionalities have been wrapped into a single IP core interfacing with a single external memory. Integration of PTME into SCoC has been performed without any major problem, but the PTME internal design contains the worst case combinatorial path limiting the overall performances of the design.

Description of the IP (from RD3)

The Packet Telemetry Encoder (PTME, see Figure 15) synthesizable VHDL model comprises several encoders and modulators implementing the Consultative Committee for Space Data Systems (CCSDS) recommendations and the European Space Agency (ESA) Procedures, Standards and Specifications (PSS) for telemetry and channel coding. The European Cooperation for Space Standardization (ECSS) documents will be based on the CCSDS recommendations, permitting the PTME to be used in future projects. The PTME model is based on the knowledge and experiences from the preceding VCA, VCM and RESCUE standard devices.

Available documentation

In addition to a Product Brief Sheet, the PTME documentation consists in 2 files :

- “Packet Telemetry Encoder (PTME) VHDL Model” (RD3) provides a full description of the PTME
- “Spacecraft-Controller-on-a-Chip adapted Packet Telemetry Encoder VHDL Model (SCoC_PTME)” (RD4) provides a description of the PTME configuration for SCoC

Success

The structure of the PTME testbench, which is based on a library of procedures and functions, allows re-using this library for the top-level (SCoC) test bench with very few tailoring. This allows focusing on the development of the test vectors instead of the development of the testbench structure.

The completeness of the documentation also allows easily using and developing test vectors for the PTME.

Problems encountered

The main constraining point was related to the baud rate generation for PacketAsynchronous interfaces. Indeed, the baud rate is dependant on the clock frequency of the system, which is hardware defined in the PTME. Thus it becomes difficult to tune the system performances without re-synthesis of the system (to adapt to the targeted frequency).

Another point is that the PTME performs 8 bits access on the AHB bus, then (on a 32 bit bus) occupying four times the bus bandwidth compared to its actual requirements.

Possible evolutions

The main required evolution for integration of the PTME in a complex SoC is to reduce the AHB bus bandwidth utilization by performing 32 bits access when possible.

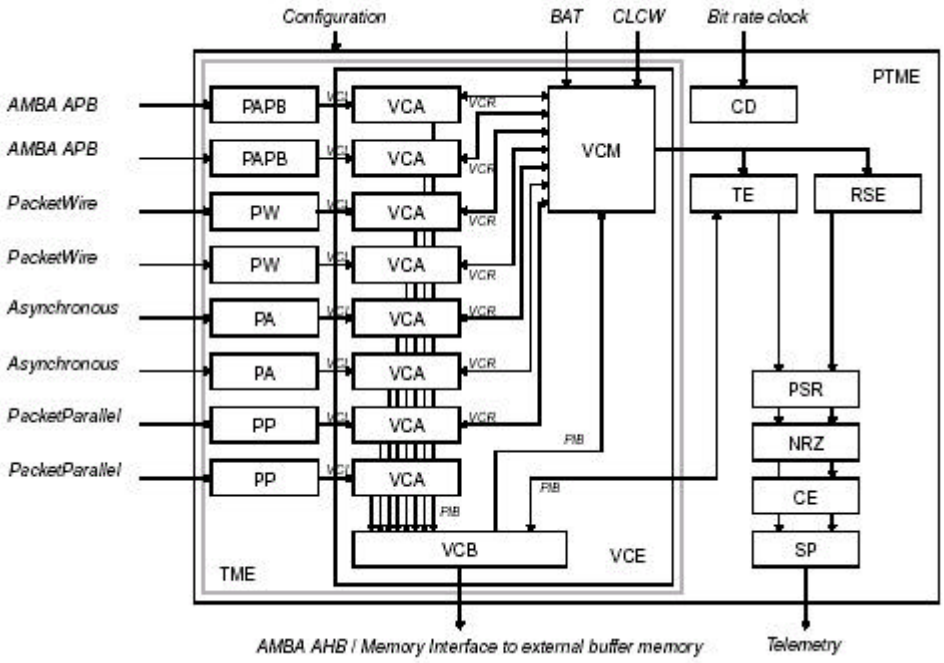

Figure 15 : PTME block diagram (example from RD3 with eight input interfaces)

The purpose of the Packet Telemetry Encoder *(PTME) synthesizable VHDL model is to provide the user with a single module implementing the* Consultative Committee for Space Data Systems *(CCSDS) recommendations for telemetry and channel coding.*

5.2 IP PROVIDED BY VENDORS

5.2.1 PCI bridge

The PCI bridge integrated into the SCoC is made of 2 parts : a PCI core handling PCI protocol corresponding to the InSilicon (now Synopsys) PCI core and a ESA developed PCI wrapper providing AMBA AHB and APB interfaces to this commercial core (see Figure 16). The PCI interface has been evaluated by performing exchanges between 2 BLADE boards, the PCI being monitored by a commercial PCI analyser from VMETRO. The PCI core has been implemented as a black box into the SoC.

Description of the IP

The PCI core provides full 32 bits 33 MHz PCI functionalities in host and satellite mode. Its AMBA interfaces allow performing DMA in Master or Target mode. The PCI host bridge is associated to a PCI Arbiter developed by ESA.

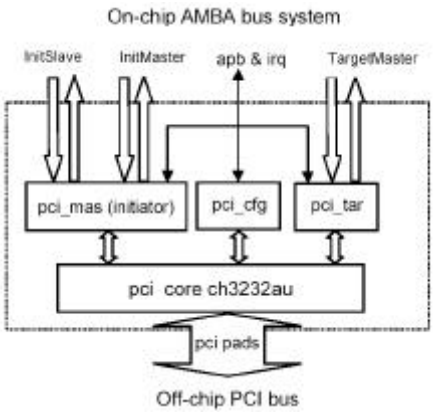

Figure 16 : AMBA to PCI interface (from RD5)

The PCI bridge is composed of a commercial PCI core wrapped into a structure providing standard AMBA AHB and APB interfaces.

Available documentation

The documentation available at Astrium for the integration of the PCI into the SCoC was limited to the PCI wrapper documentation “PCI TO AMBA BRIDGE VHDL MODEL DATASHEET” (RD5). The documentation of the PCI core was not available, as it has been integrated as a gate level netlist black box.

Success

Thanks to the work performed by ESA on the PCI wrapper, the PCI bridge was easily integrated into the SCoC. A basic PCI emulator has been developed to test the integration into the design at RTL level. Basic hardware evaluation was successful at first power up of the system.

Problems encountered

The main problem was contractual and related to the use of the PCI core for the SCoC activity. As the PCI core is property of a third party, the finding of a contractual solution for using the PCI core for an R&D study has been painful and the transfer of this IP from ESA to Astrium had to be restricted to compiled netlist and encrypted simulation model.

The integration of a compiled netlist for synthesis has been very straightforward while while the tools used integrate support for this methodology

From a technical point of view, when 2 BLADE boards try to perform extensively DMA exchanges from each other on the PCI bus, it was appearing unwanted abort of some transfers (details in RD23). The problem disappeared while inverting priority on the AHB bus between the Initiator and the Target interface. This problem revealed another problem related to AHB bus arbitration mechanism: the missing latency control.

The transfer performances between two BLADE were relatively average (see Figure 17), mainly due to the particular case of implementation of SCoC in an FPGA, leading to a system frequency lower than the PCI frequency.

Possible evolutions

The development of a full VHDL PCI core, available for all the European space community, for complex host bridge and simple PCI agents would help to get independence for the development of ASIC integrating PCI functionality.

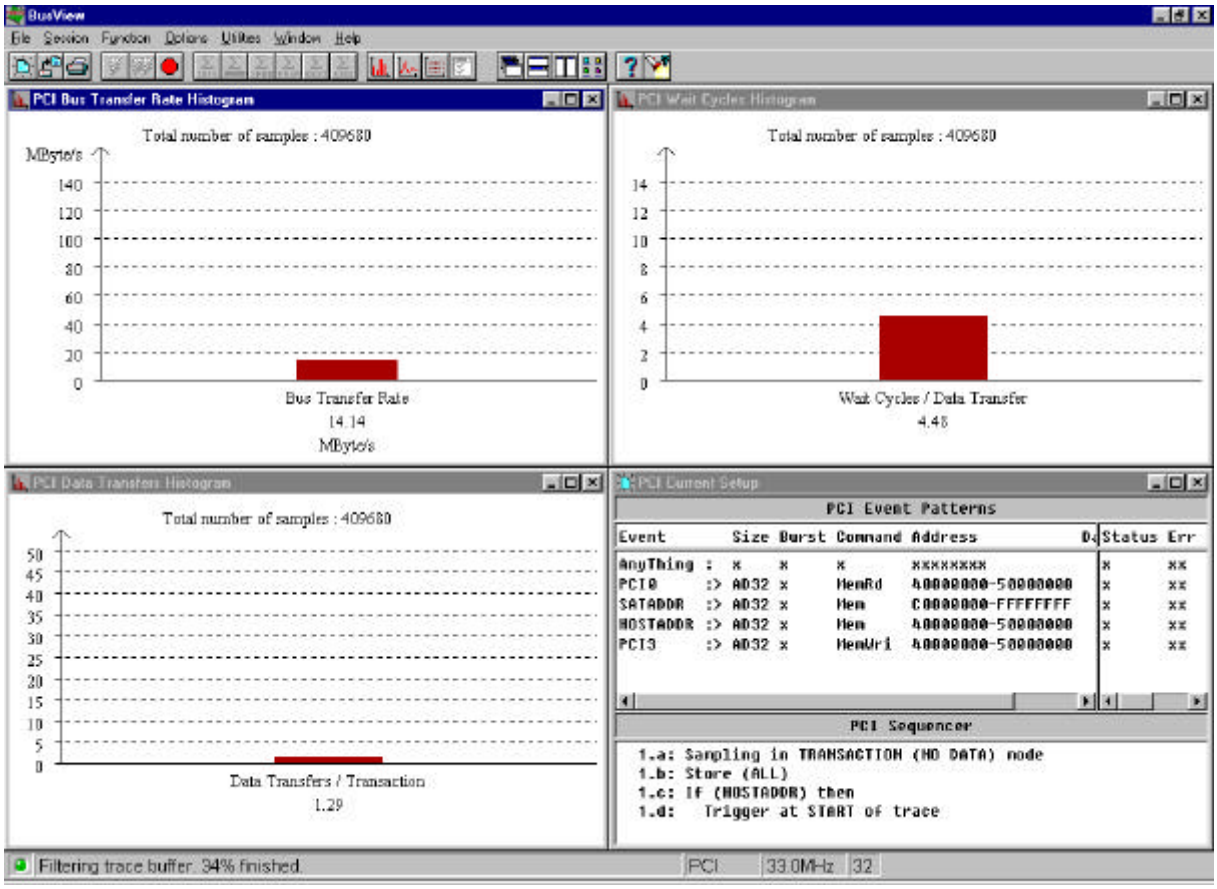

Figure 17 : Performance on the PCI bus for exchange between 2 BLADE boards

Bus performance analysis for 2 board performing symmetric DMA transfers on the PCI bus are relatively average, in terms of data rate (14.14 Mbytes/s) burst length (most of transfers are 2 words length) and wait states.

5.3 IP FROM THE ASTRIUM ASIC BACKGROUND

5.3.1 PTCDIP : Packet Telecommand Decoder

The PTCDIP has been developed from the PTCD ASIC design (which is a transfer from the GPS PTD ASIC to the MITEL PTCD ASIC). The VHDL code of this ASIC has been modified to integrate AMBA AHB and APB interfaces, making it suitable for integration in a SoC. In order to preserve a synchronous design, the ASIC code has been modified instead of developing a wrapper (see Figure 18).

Description of the IP

The PTCDIP integrates all the decoding and verification stages of the CCSDS Telecommand protocol from the coding layer up to the transfer layer. The IP interfaces with an external SRAM and PROM through an AHB interface. It can be configured through an APB interface. The received segments can be transferred either serially through pre-decoded MAP interface, or to the CPU, which is informed of the reception of a valid TC segment by an interrupt. The PTCD can generate Telemetry packets reporting its internal state.

Available documentation

The PTCDIP is described by 2 documents, which are not self-contained, as the telecommand protocol is not described. The PTD-ASIC datasheet is necessary to use the PTCDIP from an external point of view. The documents specific to the PTCDIP are the following :

- "Packet Telecommand Decoder VHDL Core Specification ICD" (RD6) provides the specification of the modified parts (between the PTCD and the PTCDIP) the HW/SW ICD and an overview of the architecture
- "Packet Telecommand Decoder HW User's Manual" (RD7) provides installation of the VHDL structure, and the associated testbench and is dedicated to implementation into the SoC.

Success

The PTCDIP is based on the re-use of a flight proven design, the core functionality (handling the TC protocol) has not been modified, and then the verification/validation of the IP is facilitated compared to a full re-design of the IP. No mal-function related to the coding layer or the transfer layer has been found.

Problems encountered

The PTD design has been performed in the early 90s, and the design integrates optimisations that were necessary to fit into the selected gate array. These optimisations made the modification of the memory interfaces for the IP design more complex.

Possible evolutions

The PTCDIP can be modified to integrate upper level services, such as extended CPDU capabilities, external authentication unit.

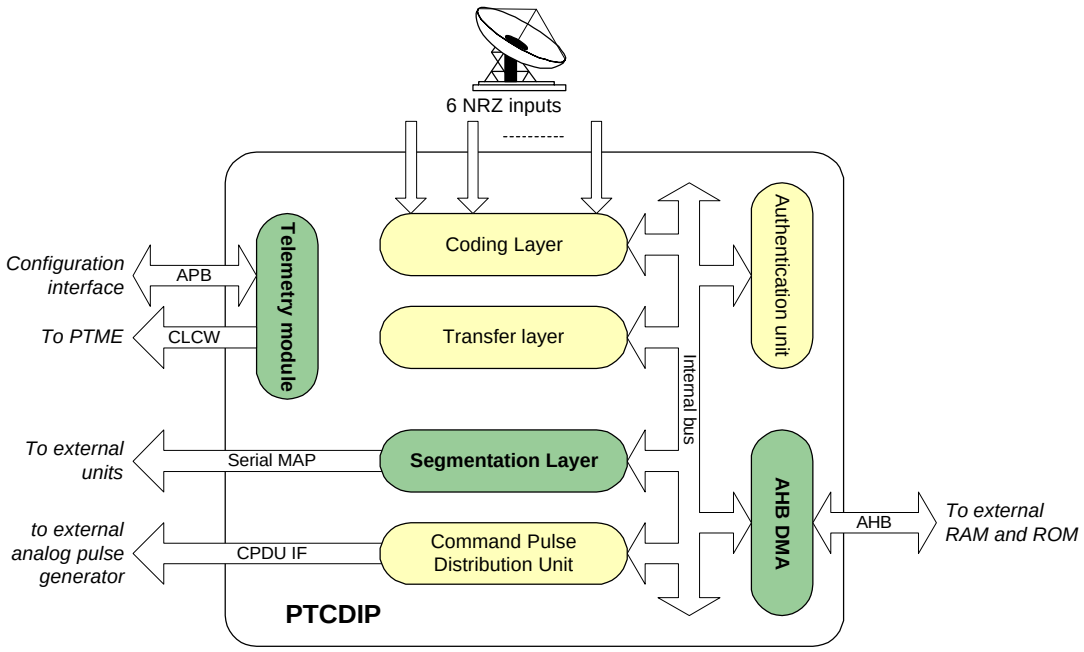

Figure 18 : Modifications from PTCD to PTCDIP

The modifications of the ASIC design were concentrated into 3 blocks (dashed on the figure). The core blocks handling TC protocol (plain yellow blocks) are not modified, thus avoiding insertion of bugs in the protocol management functionality.

5.3.2 IP1553 : Mil-Std-1553 controller

The IP1553 design is based on the former MAT53 Astrium ASIC, which integrates the Bus Controller, Remote Terminal and Bus Monitor capabilities on the Mil-Std-1553 bus. The IP1553 design is based on the reuse of core functionalities of the ASIC and a modification of the user interface for compatibility with the AMBA AHB and APB busses (see Figure 19).

Description of the IP

The IP1553 is based on coding and protocol layers designs already integrated and validated through numerous applications. The user interface is modified to comply with the requirements of a VHDL IP : AMBA AHB and APB interfaces, interrupt services. The design is also modified in terms of performances for use on a 32 bits data bus.

Available documentation

The “IP1553 Specification” (RD8) describes the IP functionalities and the interfaces (pin and registers).

Success

The IP1553 has been developed for internal uses and has been easily integrated into SCoC after minor modifications of the user interface for compatibility with the AMBA AHB bus. The testbench structure at IP level has been re-used and no VHDL code has been written for the simulation of the IP1553 integrated into SCoC.

The IP1553 was fully operational for BC and RT modes at first implementation into the hardware, showing the effectiveness of the reuse methodology.

Problems encountered

No problems were encountered related to the IP1553, as the block was already validated as re-usable IP for another project.

Possible evolutions

As the overall BC/BM/RT functionality is not required for all developed system, an RT only Mil-Std-1553 IP has been developed to be integrated into a complex system or to be used in a standalone mode into a single ACTEL RT54SX32S FPGA.

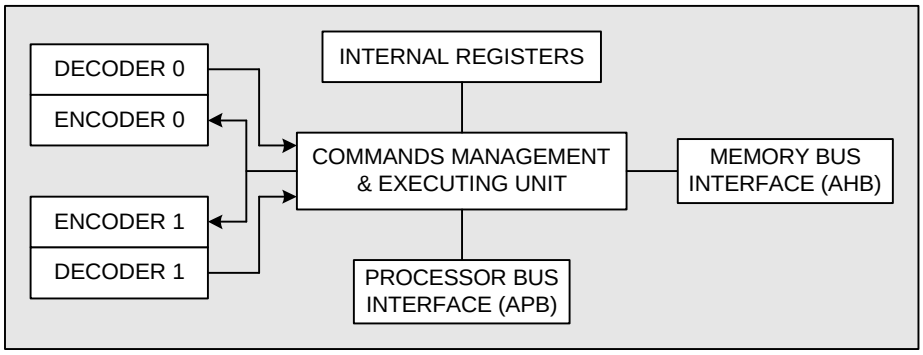

Figure 19 : Block diagram of the IP1553

Most of the modifications from the MAT53 design to the IP1553 design take place in the "Memory Bus Interface" block and the "Processor Bus Interface". The parts of the IP handling the Mil-Std-1553 protocol have not been modified, reducing the risk of bug insertion.

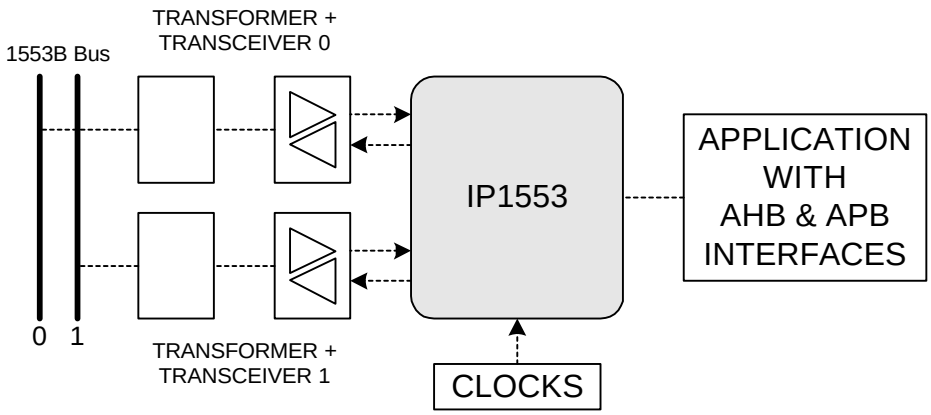

Figure 20 : IP1553 environment

The integration of the IP1553 in a system is really simple as the external interfaces are very limited. The Bus Controller function integrates a programmable sequencer whose program is stored in the external memory accessed through the AHB interface.

5.4 NEWLY DEVELOPED IP

5.4.1 HKPF House Keeping "Packetizer" function

The HKPF function (see Figure 21) automatically generates telemetry packets transferred to the virtual channel 0 of the PTME. The telemetry sources are internal – telecommand report, time packet, context RAM dumping - for the SCoC design study, but the design can be upgraded to integrate automatic analog and digital status acquisition for example.

Description of the IP

The HKPF function has been specifically designed to interface with the CTM APB Time packet interface, the PTCD status report interface and the PTME Virtual Channel 0 input. Telemetry packets are transferred from the CTM or PTCD to the PTME when receiving a pulse command or register programming. In addition, HKPF integrates a Context RAM buffer management that can be filled through the APB interface and that is dumped to the PTME when receiving a pulse command or register programming.

Available documentation

As the size of the IP is relatively small, a single document groups all information relative to the HKPF: "HouseKeePing Function VHDL Core Description" (RD11). The document contains the functional description, an overview of the architecture, and all information relative to the implementation of the block into an upper level design.

Success

The function is working correctly according to its specification. The internal architecture of the HKPF allows re-using the basic functions to develop a more complex Automatic House Keeping Manager.

Problems encountered

No specific problems were encountered during the development of the IP. The specification of the IP, along with the development of standard requirements for such a function, was out of the scope of this activity. That led to develop a basic housekeeping function to validate the opportunity of such a module into the SCoC.

Possible evolutions

The HKPF design can evolve to introduce more complex house-keeping functions, such as the management of analog or digital acquisitions, the connection to the AHB CPU bus as a master in order to retrieve various telemetry information in SCoC (and even externally through the PCI) The integration of a micro-controller would allow the automatic construction of complex telemetry reports. See Figure 22.

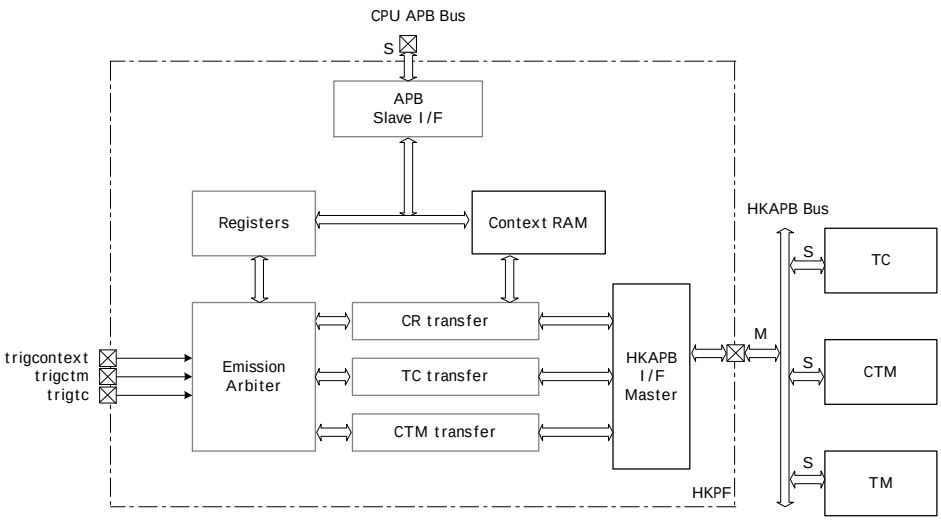

Figure 21 : HKPF block diagram

The HKPF functionalities can evolve by integrating different blocks in parallel of the "Context RAM transfer", "TC transfer" and "CTM transfer" functions.

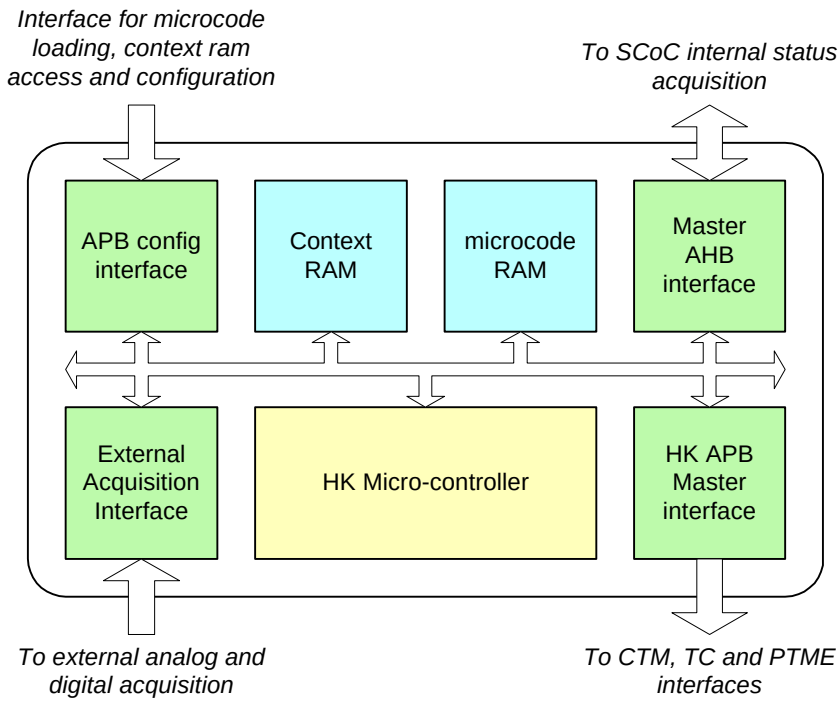

Figure 22 : Possible evolution of the HKPF function

The embedded micro-controller allows generating complex telemetry report based on acquisition of external status (analog and digital), internal register or memory content, in parallel of existing interfaces with CTM and TC. The statuses are directly sent to the PTME through the HK APB interface.

5.4.2 AMBA small IP

*Small IP specified from the SCoC functional specification has been developed to enhance the SCoC internal architecture. These blocks are the A**H**B **D**irect **M**emory **A**ccess (HDMA, see Figure 23) function and the IO memory controller function (IOMCTRL, see Figure 24).*

Description of the IP

The HDMA is an interface allowing DMA transfer between 2 AHB busses. It is used into the SCoC for data transfers between the CPU AHB bus and the IO AHB bus. The HDMA integrates a FIFO to handle functionally asynchronous access to the 2 busses (i.e to decouple the access to the two busses).

The IOMCTRL is a dedicated IO memory controller that embeds hard-wired memory area protection, in such a way that parts of memory are only accessible by specific AHB masters. This functionality is useful to protect the telecommand memory content (authentication keys for example) from corruption by other blocks on the same bus (1553 and HDMA have possibility to address the whole memory).

Available documentation

Each IP is described by a single document providing all information: Specification, HW/SW interface, architecture. These documents are “HDMA Specification and Architecture” (RD12) and “IOMCTRL Specification and Architecture” (RD13).

Success

The HDMA is configurable in order to limit the burst length on the AHB busses, allowing a control of the bus latency at system level, especially for the IO bus.

The IOMCTRL protection areas are configurable at hardware level, so that the design can be easily tailored for specific applications.

Problems encountered

The HDMA functionality is limited to only DMA transfers, and its usage is not efficient for transfer of small data structures.

Possible evolutions

An evolution of the HDMA design would be necessary to enhance the performances of the whole system. Examples of possible evolutions are :

- management of multiple DMA channels,
- bandwidth allocation (synchronisation on CPU real time slots),
- integration of bridge functionalities, to perform direct transfers between 2 busses,
- DMA transfers on the same bus.

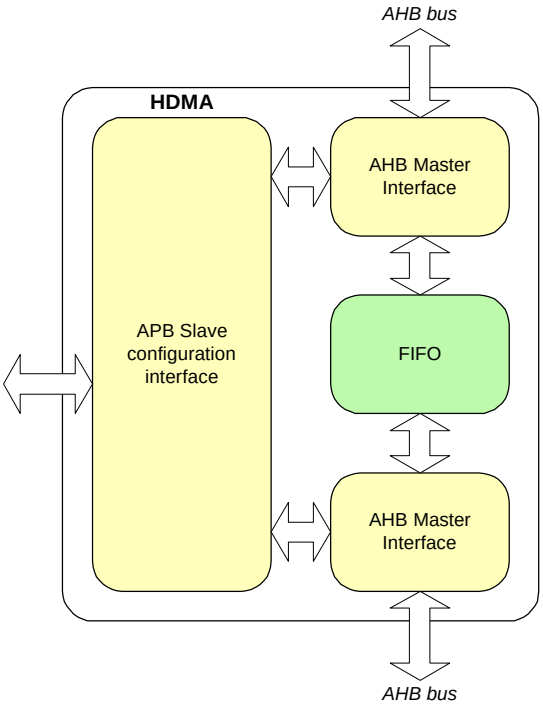

Figure 23 : HDMA block diagram

The structure of the HDMA is symmetric. the same block is re-used for both AHB busses. The HDMA is configurable - FIFO size, maximum burst length – to tailor the design to specific system requirements

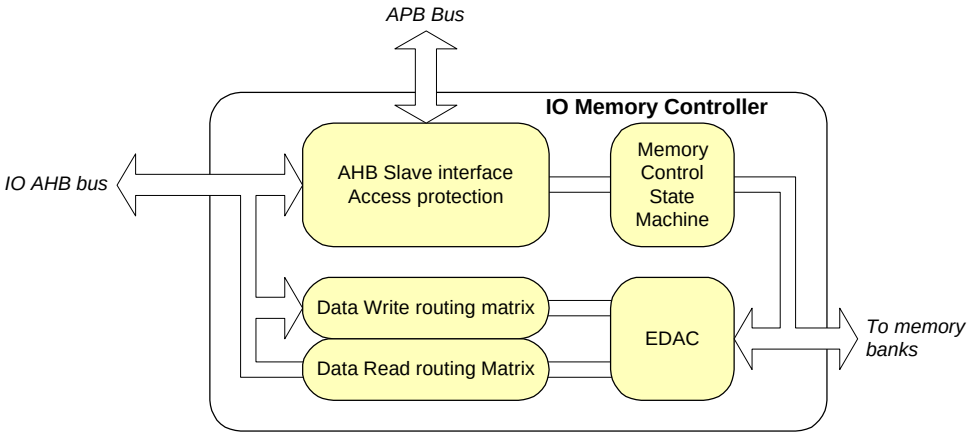

Figure 24 : IO Memory Controller Block Diagram

IOMCTRL integrates access protection based on the master number. This allows protecting the RAM area dedicated to PTCD from 1553 accesses for example.

5.4.3 Spacewire

The Spacewire IP core (Figure 25) allows achieving transfer rates at up to four times the system frequency. The VHDL structure of the IP core clearly separates the Spacewire protocol management from the host interface, allow re-using either the full IP or only the core. The host interface provides intelligent DMA services that can be tailored to be adapted to specific system requirements.

Description of the IP

The Spacewire IP is divided in two parts. The Spacewire core handles all the requirements of the ECSS specification (AD5), up to the FIFO interface described in the document. The host interface wraps the Spacewire core providing AMBA AHB and APB interfaces and also services such as linked list of packet for transmission and double buffer management for reception.

Available documentation

"Spacewire IP Core Specification and Architecture" (RD9) provides the specification and functional description, the HW/SW ICD and an overview of the architecture. "Spacewire IP Core HW User's Manual" (RD10) is intended for users that would like to use the block. It provides information on the VHDL structure, the adaptation of the block for different technologies, the execution of the simulation and synthesis of the standalone block.

Success

Spacewire Testbench developed by Austrian Aerospace for ESA has been re-used. These simulations were used to design and validate the Spacewire protocol core of the IP.

The implementation of the IP into SCoC has also been validated through the use of 3 BLADE boards for the A3M project. Spacewire links were used to establish communication between the 3 CPU nodes.

Finally, measured performances on the hardware indicate that the implementation of the IP into the SCoC structure does not limit the intrinsic maximum achievable data rate on the interface.

Problems encountered

Problems were encountered during the gate level design around the Spacewire IP core. Indeed, the XILINX tool used was not routing the reception stage properly (reconstructed clock and shift register) with only high-level timing constraints, and then mal-function were detected on the first hardware. The solution was to pre-place some parts of the reception interface "manually" (a constraint file is written for placement). This problem is mainly linked to the targeted technology.

Possible evolutions

The developed Spacewire IP core does not corresponds to the last ECSS specification (AD5), as this latter was not finalized at the time of development of the IP. It is possible to modify the IP to comply to the last version of the specification.

Depending on system requirements, it is also possible to tailor the host interface to enhance the performances and the provided services.

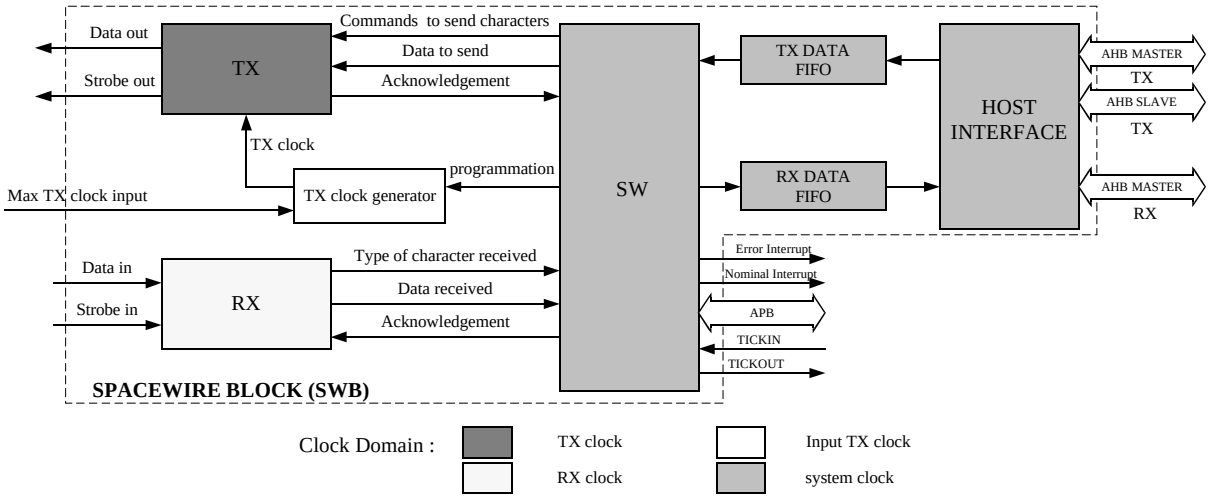

Figure 25 : Spacewire IP core structure

The host interface is completely separated from the blocks handling the Spacewire protocol, so that it is easy either to reuse the protocol core or to tailor the host interface for specific system requirements

Link frequency (MHz)	40	80
Packet size (bytes)	40	40
useful data in the bit stream (%)	79,2%	79,2%
Calculated Max data rate (Mbits/s)	31,680	63,360
Measured Max data rate	31,68	63,36
Performance ratio (measured/calculated)	**100,0%**	**100,0%**

Figure 26 : Comparison of measured and calculated Spacewire performances

The performances in DMA transfers of 1 Mbytes of data have been measured at different frequencies with system clock frequency of 20 MHz. The Spacewire IP transmits and receives data at 100% of calculated maximum rate.

Page intentionally left blank

6 ASSESSMENT OF THE DESIGN METHODOLOGY

6.1 INTRODUCTION

The particularity of the development of a SoC comes principally from the use or re-use of blocks coming from different sources. Each block has its own particularities. The different phases of the development of an ASIC have been performed for the SCoC development. Limitations of the design flow as described in AD3 for the development of the SCoC are established and an adaptation of the methodology to the SoC design is proposed.

The development of a SCoC is based on the use of IP cores whose status is different from one block to another. Blocks are either newly developed for the application, come from previous ASIC developments or are well proven in a commercial application. The complete validation of the block functionalities when integrated into the SCoC with traditional methods such as simulation is not possible due to either the difficulty of generating pertinent test vectors at IP core interface level or due to the duration of such simulations.

In addition, most of the IP cores used into the SCoC are highly configurable and the number of possible combination of parameters is enormous, so that only a subset of configuration is effectively validated for the IP.

The methodology is based on a complete verification by simulation of the standalone IP core configured for the SCoC, and the progressive integration of IP cores within the global design, with, for each block newly integrated, the development of specific test vectors to validate the integration of the functions into the system. To complete the validation of the integration of the IP core into the design, a prototype board is used to speed-up the verification (compared to simulation times). Targeting a prototype board with FPGA also verifies the synthesizability of the design.

The next sections describe, for each phase of the ASIC design, the actual method used for the development of the SCoC and focuses on the positive and negative points of the phase. The phases analyzed are the “Initial analysis” and in particular the functional specification and feasibility study phase, the “Architectural Design” and in particular the re-use of test vectors and test structures from IP verification to system verification, the “Gate Level Design” to some extent as this phase has been executed only for targeting a prototype FPGA, and the “Hardware Evaluation” that stands for a partial validation of the design. The last design phase analyzed is the development of an IP core as experimented with the development of the Spacewire block.

6.2 INITIAL ANALYSIS

The functional specification of a SoC design cannot be a self-contained document as it is based on the use or reuse of existing IP cores. Initial analysis should follow the same concept of reuse by the document and information. That means that the information necessary for feasibility study (complexity …) are presented for each IP core in a re-usable form. Feasibility study is limited here to the analysis of the design complexity as the foundry selection was out of the scope of this activity.

Description

The functional specification cannot contain all the information relative to each block implemented within a system on a chip. Then, for each block implemented, a reference to the specification of these blocks is associated to the selected configuration if relevant. For example, all the requirements of a Sparc V8 CPU core are not re-written into the functional specification, and the document gives a reference to the LEON1 document, along with a description of the configuration of LEON specific for the SCoC requirements.

As the ASIC foundry was not foreseen in the scope of this activity, the ASIC feasibility study relative to foundry has not been performed. The study has been assessed in terms of complexity (gate count), pin function, and internal data path performances.

The feasibility study extremely relies on the availability of information on the implemented blocks, in particular for a gate count. Nevertheless, as blocks are often highly configurable, it is not always possible to get accurate information for a specific use. Meanwhile, it is necessary to facilitate the re-use of information relative to the block to establish the feasibility of a SoC

Success

The feasibility study has been performed by developing EXCEL sheets grouping all relevant information for each block (see Figure 27). In particular, the sheet integrates a gate count for particular technology, an external pin count, a reference to the documentation and a comment on the use of the block. A sheet may exist for each configuration of a particular block. The sheet is updated at the end of the design to centralize accurate information that will be available for future developments.

Problem encountered

At the time of the edition of the functional specification, information was not available for each block. Then the HW/SW ICD only provides a mapping of the blocks on the busses, without detailing the register content for each block. The spread of information reduces the use of this document for the development of SCoC test vectors, in particular for the writing of the CPU software. This situation is balanced by the development of basic software library for each block, abstracting the register level.

BLOCK NAME	SPACEWIRE	**VERSION**	1.0
DESCRIPTION	SPACEWIRE interface with FIFO and AMBA application interface. Can be master or slave in the AHB bus. Compatible with version DRAFT-1 of SPACEWIRE specification		
DOCUMENTATION	SPACEWIRE IP CORE SPECIFICATION AND ARCHITECTURE		
OBSERVATIONS	Developed by ASTRIUM Layout of the reception stage to be done carefully (reconstructed clock and shift register)		

BUS DEFINITION	**Value**	**Comment**
AHB MASTER		
HRESP SPLIT supported	YES	
HRESP RETRY supported	YES	
HRESP ERROR supported	YES	
HSIZE values used	32	
Configurable base address	NO	Spacewire can access full address range
Maximum latency requested		No request, depending on throughput
Maximum burst size	2	
Idle inserted between bursts	NO	
Bus throughput (Maccess/s)	5	Maximum at 100MHz RX and TX
Requested external memory size (Kbytes)		No static request
AHB SLAVE		
HRESP SPLIT generated (precise conditions)	YES	When Spacewire FIFO and Host FIFO are full
HRESP RETRY generated (precise conditions)	NO	
HRESP ERROR generated (precise conditions)	YES	in case of read access
All HSIZE values supported	32	
Typical wait states (Nb CLOCK)	0	
Maximum wait states (Nb CLOCK)	4	
Supported throughput (Mbytes/MHz)	0,4	Maximum at full speed TX
Requested address space (Mbytes)	0	1 word minimum
APB SLAVE		
Requested address space (Bytes)	64	Aligned on a 64 bytes boudary

EXTERNAL I/O	**COMMENT**	**SE**	**DIFF**	**TOTAL**
INPUTS		2	0	2
OUTPUTS		2	0	2
INPUTS/OUTPUTS		0	0	0

		XILINX			
BLOCK AREAS	COMMENT	LUT	RAMBLK	DFF	IO
SPACEWIRE		2000	4	1144	8
TOTAL AREA		2000	4	1144	8

Figure 27 : Spacewire information sheet

For each block, an EXCEL sheet provides all information necessary for the feasibility study : characteristics on the AMBA AHB and APB busses, number of external I/Os and gate count for a particular technology.

6.3 ARCHITECTURAL DESIGN

The architectural design phase integrates the architecture of the SCoC along with the detailed specification of the IP cores to be developed. The detailed architecture of each block is provided in block specific documents. Then the SCoC architectural design document references documentation of each block. For the simulations, the main point is the re-use (and eventually porting) of simulation environments from the IP core up to the system level.

Description

At system level, i.e. SCoC level, the architectural design document is concentrated on top level issues, such as the internal bus architecture, the clock and reset distribution, the specific tailoring of each implemented block, the connection of each block to the internal busses and services (such as interrupt or synchronisation pulses). The glue logic is also described in detail.

The testbench structure (Figure 28) and the simulation plan are grouped in a single document. The SCoC testbench structure is based on the re-use of the IP core testbenches as far as possible. This is made possible by the development of modular testbenches, based either on function libraries such as the PTME or emulators for EADS-Astrium developed blocks.

In parallel of the architecture design at system level, the architectural design of each block to be developed is performed, almost as a separate activity. Blocks are integrated progressively in the SCoC. Simulation test suite is also developed progressively. This implies that the SCoC shall be configurable at least for the presence or not of the block.

Success

The progressive integration of the blocks into the SCoC facilitated the testbench development and the simulation of the system. Indeed, bugs discovered were easily circumvented to the added block at a particular step.

Except for the top-level VHDL file structuring the SCoC testbench, only a few VHDL code has been developed specifically for the top-level simulation. All external interfaces of the IP cores were handled by standalone emulators, which have been re-used at top level.

Finally, the development of standalone AMBA emulators reduced the need of specific development for the testbench of each block.

Problem Encountered

The main problems encountered during this design phase were related to the LEON configuration structure, that does not separate LEON core functionality configuration from the AMBA busses configuration. LEON structure has been partially broken in order to re-use elementary functions.

While test structure (for external signals) is fully re-used at top level, internal test vectors of IP cores must be converted. For example, APB interface is handled by an emulator at IP core level while it is handled by LEON software at SCoC level.

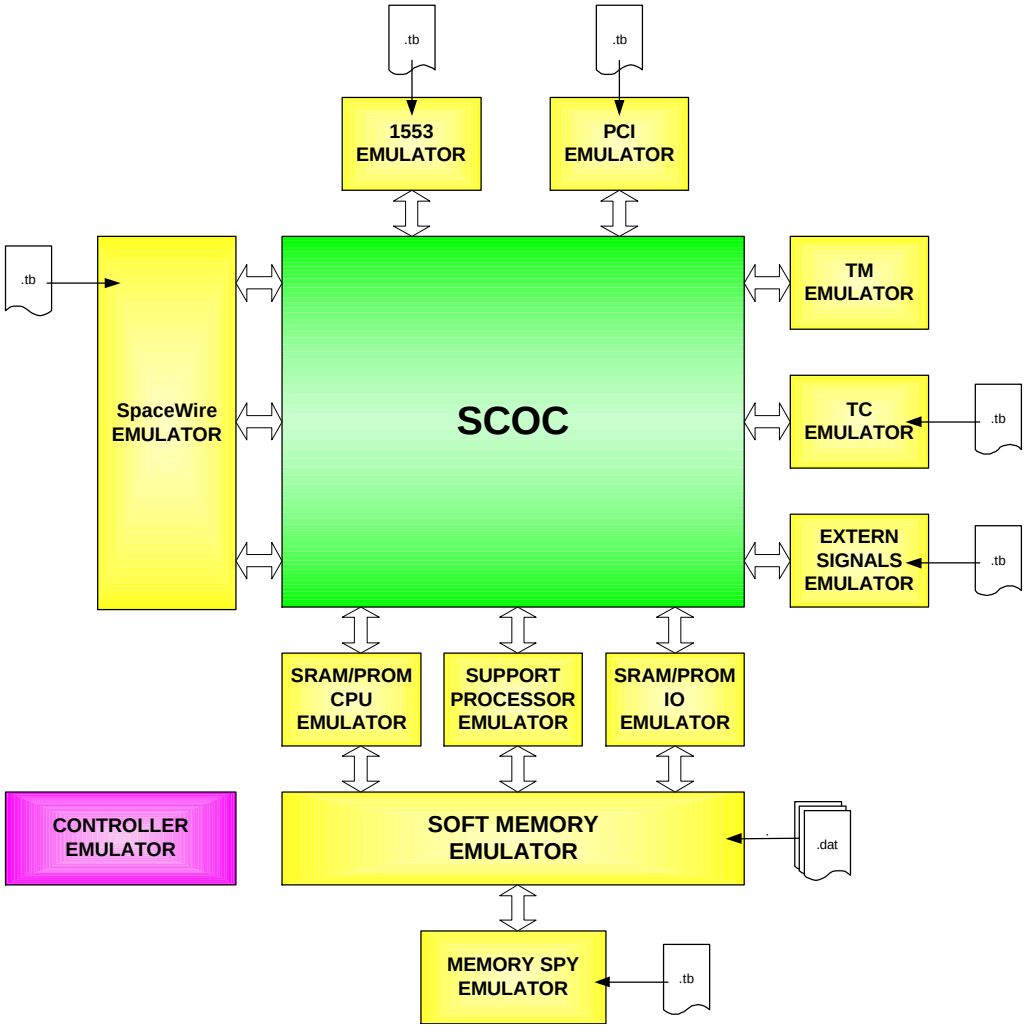

Figure 28 : Overview of the SCoC testbench structure

Most of the emulators composing the SCoC testbench are taken and re-used from the IP core testbench. The standardization of emulators allows inter-operability at upper level. The TM emulator is in fact an instantiation of functions provided with the PTME IP. The controller emulator allows synchronisation of the emulators altogether.

6.4 DEVELOPMENT OF AN IP CORE

The Spacewire IP Core has been developed to provide an AMBA Spacewire bridge suitable for use in SoC applications. It is also an example to study the development methodology of a re-usable IP core.

The development of an IP core suitable for integration into a SCoC starts by its specification. Functional Requirements for the Spacewire IP Core are of 3 types :

- Requirements associated to the external protocol (AD5)
- Requirements associated to the standardized interfaces : AMBA bus standard, HW/SW interface for interrupt management.
- Requirements associated to HW added services : management of linked lists of packet, double buffer.

The Spacewire IP-core specification is mainly related to the services specification as the Spacewire protocol specification is an external document and the internal bus specification is described in the SCoC functional specification. For the Development of an IP, the specification must be compatible to external standards in order to ensure a high potential of re-use of the block.

The Architectural Design follows a methodology ensuring coherency between IP cores : selection of a library name for the IP, use of central technology specific libraries (for example for RAM instantiation), development of testbench according to specific requirements (re-usable testbench structure). This eases the integration of multiple IP into the SCoC.

Synthesis is performed at IP level to ensure that the design can be synthesized and to find out the worst-case combinatorial paths. It is important to retrieve and to check this information at IP level as the task is much more difficult at system level.

The documentation delivered for the IP contains all information to :

- configure the IP core for a specific application : List of configurable parameters
- simulate this configuration at IP core level : Testbench description
- implement the IP core into a system : VHDL structure description
- use the IP core at system level : Functional specification, HW/SW ICD.

In addition, the documentation (delivered or not depending on the licensing mode of the IP) allows the maintenance of the IP core by providing an architecture description.

6.5 GATE LEVEL DESIGN

The gate level design has been performed targeting a commercial VIRTEX-E FPGA intended for function prototyping. Nevertheless, this step allows to reveal potential problems linked to synthesis of the system. As the main limitation was timing performance, only global synthesis has been performed, allowing the tool to perform global optimisations.

Description

The synthesis of SCoC has been performed with the Synplify tool targeting a XILINX VIRTEX-E FPGA. Scripts for synthesis with the compilation order of each IP core can be used either for local synthesis or SCoC synthesis, thus avoiding errors related to script writing. Constraints have been set in synthesis constraint file rather than in the VHDL code in order to be able to re-use the VHDL design with different synthesis tools. Nevertheless, this is not always possible with the used tools and some synthesis attributes remain in the VHDL code.

The timing have been analysed after synthesis and XILINX layout to determine the worst-case path and the achievable performances on the prototype board.

Finally, an extract of the RTL simulation suite has been executed on the post-layout netlist.

Success

The PCI core has been integrated as a black box in the synthesis flow, mapped in the XILINX with a netlist provided by ESA, without any particular issue. Nevertheless, it was not possible to achieve the requested timing on the PCI interface, as the selected PCI core is more suited for ASIC implantation and as the SCoC FPGA was almost full. The PCI was functioning on the system thanks to the margins provided by the PCI timing specification.

Problems encountered

Most of the problems encountered were related to the XILINX ISE tool.

Use of the XILINX ISE tools revealed that the obtained results are dependant on the platform and on the usage of the platform in case of multi-user machines.

The difficulty to constrain the Spacewire RX module, along with the limitation of resources into the FPGA, leaded to a manual placement of some DFF of the design. This solution solves the problem encountered for Spacewire link and focuses on potential problem for the final ASIC layout. This constrain is reminded in the feasibility sheet of the Spacewire block (RD6) and in the Spacewire documentation (RD13)

For timing analysis, we tried the XILINX ISE to Synopsys Primetime tool link without success. The new release of ISE tools should have corrected the encountered issues.

The post layout simulation results (log files) were polluted by VITAL timing warnings that were analysed as uncritical, but whose generation could not be turned off (it is convenient to suppress the warning generation and X propagation for inter clock domain resynchronisation for example. It is not possible for some particular primitives of the Xilinx VITAL library).

6.6 HARDWARE EVALUATION

The evaluation/demonstration board BLADE has been used to test an implementation of the SCoC into a commercial FPGA – XILINX XCV2000E. Bugs discovered with this implementation, mainly located at asynchronous boundaries, have been rapidly corrected and re-validated. The evaluation also allowed measuring performances of the high speed I/Os implemented within the SCoC

Description

For the evaluation/ demonstration of the SCoC concept, a specific board – BLADE (see Figure 29) – has been developed (commercial boards were not adequate at the beginning of the activity). This specific development allowed integrating specific I/O drivers used in our space application such as Mil-Std-1553 bus transceiver and transformers

Success

BLADE has been used successfully for an evaluation of the SCoC functionalities and performances. In order to reduce the workload related to the development of test vectors, LEON simulation software has been re-used. Simulation vectors have been enhanced to take benefits of the quickness of the test execution on hardware: large transfers on high speed links, intensive tests, long sequences of telemetry.

Very little testbench is necessary to activate the board. Thanks to the integration of a dedicated test FPGA on the BLADE board, external means were limited to a Mil-Std-1553 BC/RT emulator, a PCI bus analyzer, a PC and a SUN workstation. Telecommand and Telemetry interfaces were emulated into the TEST FPGA (see Figure 30).

The XILINX Chipscope tool has been used for investigation on internal signals when necessary.

A bug on the Spacewire IP has been discovered during this phase, which would not have been discovered without intensive use of the Spacewire IP integrated into the SCoC. Then the use of large commercial FPGA for the prototyping of large or complex ASIC design (like SCoC) is fundamental.

This implementation also allowed performing test with real time software running on the CPU and using the Spacewire links and the CTM, as the board has been used for the A3M project. This test generated return on experience relative to the Spacewire HW/SW interface.

Problems encountered

The main problems encountered were linked to the limitation of the FPGA used on the board. Indeed, the full SCoC design was not fitting into the FPGA, and partial configurations were tested.

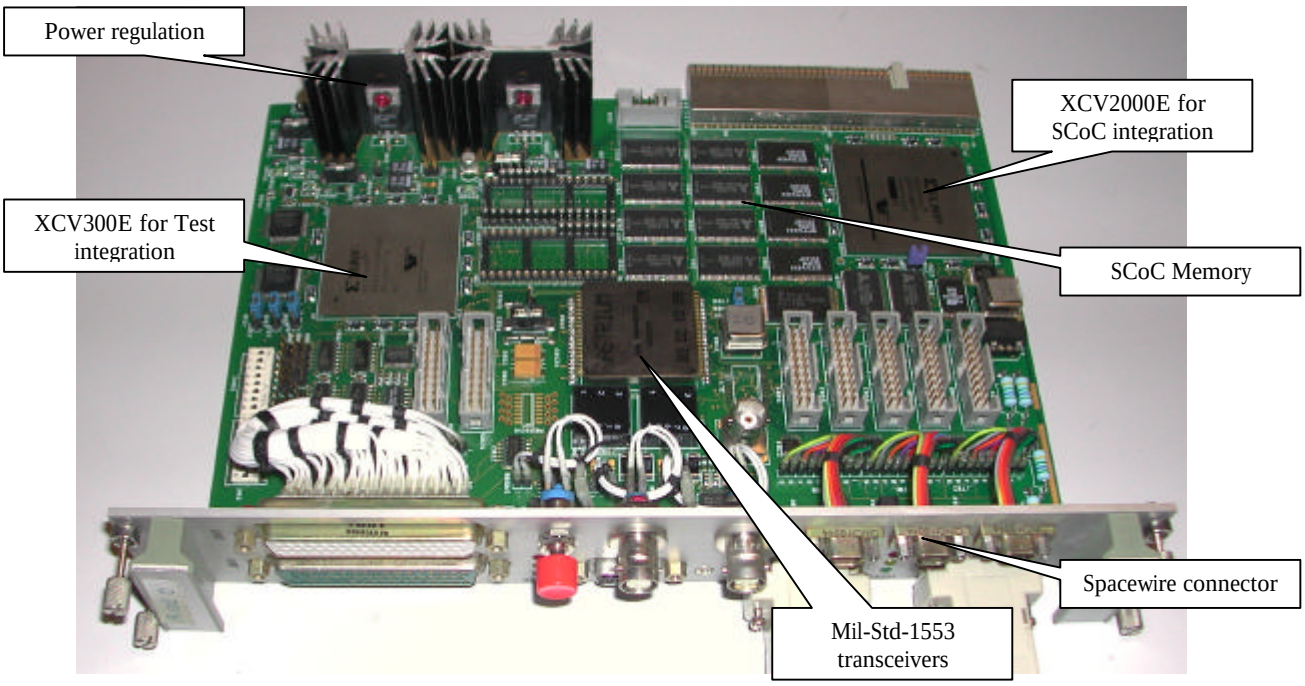

Figure 29 : BLADE board

The 6U Compact PCI BLADE board integrates the environment necessary to implement and evaluate the SCoC with a very limited test environment.

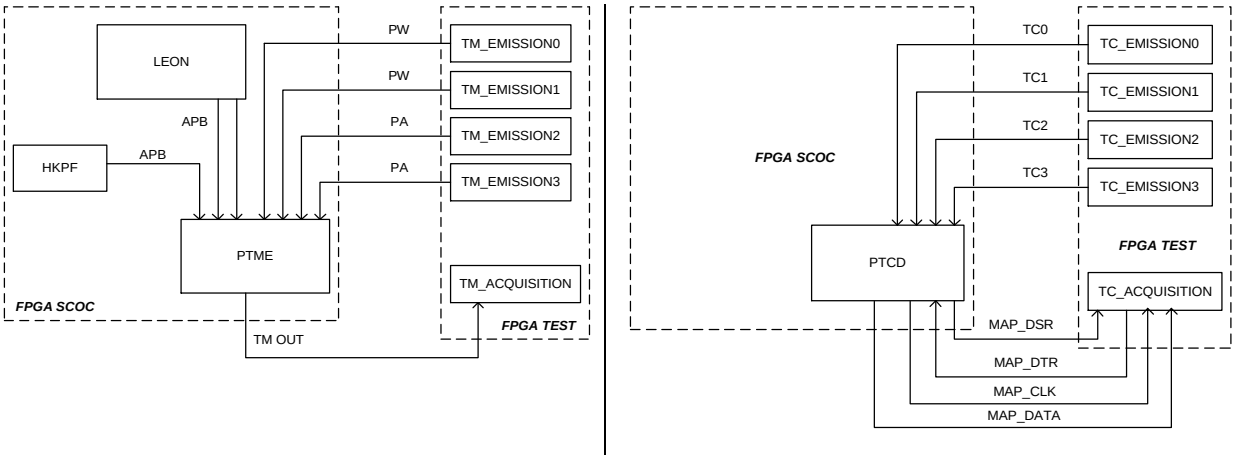

Figure 30 : Implementation of TM and TC emulators in the TEST FPGA

The TEST FPGA integrated on the BLADE board allows testing PTME and PTCD IP without external testbench development.

Page intentionally left blank

7 ASSESSMENT OF THE USED TOOLS

7.1 INTRODUCTION

The SCoC development has been using software tools for classical ASIC development. This section aims to discuss on each of these tools, mainly VHDL simulator and synthesis tool when used to develop SoC.

The main particularities of SoC designs come from the fact that SoC-ASIC may be large (> million gates) and that the VHDL description of the SoC is based on the use or re-use of blocks coming from a large variety of sources. The design of the SoC depends on the tools being able to handle these large designs and on a methodology for the use of these tools that copes with the diversity of sources.

For example, the simulation tool and simulation platform shall allow keeping the simulation times in an affordable limit. While in design phase (during which a lot of bugs are discovered and shall be corrected), the methodology used shall allow performing a complete flow (VHDL source update, RTL simulation, synthesis, P&R, hardware test) in a reasonable time. If not possible, the methodology shall allow the reduction of the number of bugs discovered at later design stages. That was done for the SCoC development by performing validation of the blocks at IP core level.

For synthesis, the software tool must be able to synthesize a system whose VHDL description is not complete (including black boxes). Most of the tools currently support this feature. For even more complex SoC, the methodology for the use of synthesis tool will shift from a global synthesis to a block-by-block synthesis with only a top-level assembly (bottom-up synthesis). Assessment on the pro and cons of each method is done below.

The hardware and software platforms used for the SCoC development is presented here below.

- **Design capture** is made in VHDL with standard text editor (nedit) running on a SUN Solaris platform. Source versioning is handled through CVS with TKCVS front end
- **VHDL Simulations** are made with Modelsim 5.7b either on SUN Solaris platform or PC Linux platform
- **Synthesis** is made with Synplify 7.21 on SUN Solaris platform
- **Place & Route** is made with XILINX ISE 5.1i on SUN Solaris platform

7.2 SIMULATION TOOL

A commonly used VHDL simulation tool used is Mentor Modelsim. This tool is well suited for verification of the SoC at RTL level. A standardization of the methodology used for the development of testbench as well as basic rules for the VHDL RTL description eases the simulation of complex SOC.

Package Libraries

For a correct use of VHDL and a correct handling of the RTL description of the SoC, common declarations must be centralized in dedicated package and libraries. In SCoC, a package is created for the AMBA bus and archived in a specific library (AMBA_LIB). The same applies to common configuration parameters (specific packages in PACK_SCOC_LIB) and common testbench parameters (packages in TB_PACK_LIB).

For example the AMBA package commonly used by all the IP cores is compiled in a specific library named AMBA_LIB (and then LEON VHDL source code has been modified to adhere to this rule). This was made necessary as VHDL compilers does not support the connexion of IP cores on the AHB bus if the AMBA package is compiled in each IP library (type IPA_LIB.amba.ahb_mst_in_type is different from IPB_LIB.amba.ahb_mst_in_type).

To cope with this library name problem, one solution would be to compile all the VHDL structures into the same WORK library, but another problem could then rise.

IP Libraries

When a lot of IP cores coming from different sources are integrated in the same ASIC to form a SoC, we have to prevent from name conflict between different IP sources. A typical example is the declaration of a FIFO entity internal to an IP. If multiple IPs declare and describe a FIFO sub-element, then the simulation tool shall be able to distinguish the FIFO description from IP X and from IP Y. Either entity names have to be made unique over the whole working library, or IP X and IP Y have to be compiled into different libraries.

To resolve ambiguity, it is recommended to define a particular library into which all packages and entities are compiled and to provide to the user a package containing only declarations that have to be visible at integration level. Name conflics are still possible, but with a limited extent, and have to be solved either by modifying the IP core or by implementing wrappers hiding IP core declaration at integration level.

Scripting

One of the advantages of the simulation tool during the debug phase of the SoC is the possibility to extend the capabilities by scripting. Thus, we have developed a set of scripts allowing to trace the CPU program execution in the wave window (with function such as step, go to …). The script is portable through different platforms. It displays the disassembled source code highlighting the line of code corresponding to the cursor position in the wave window (see Figure 31).

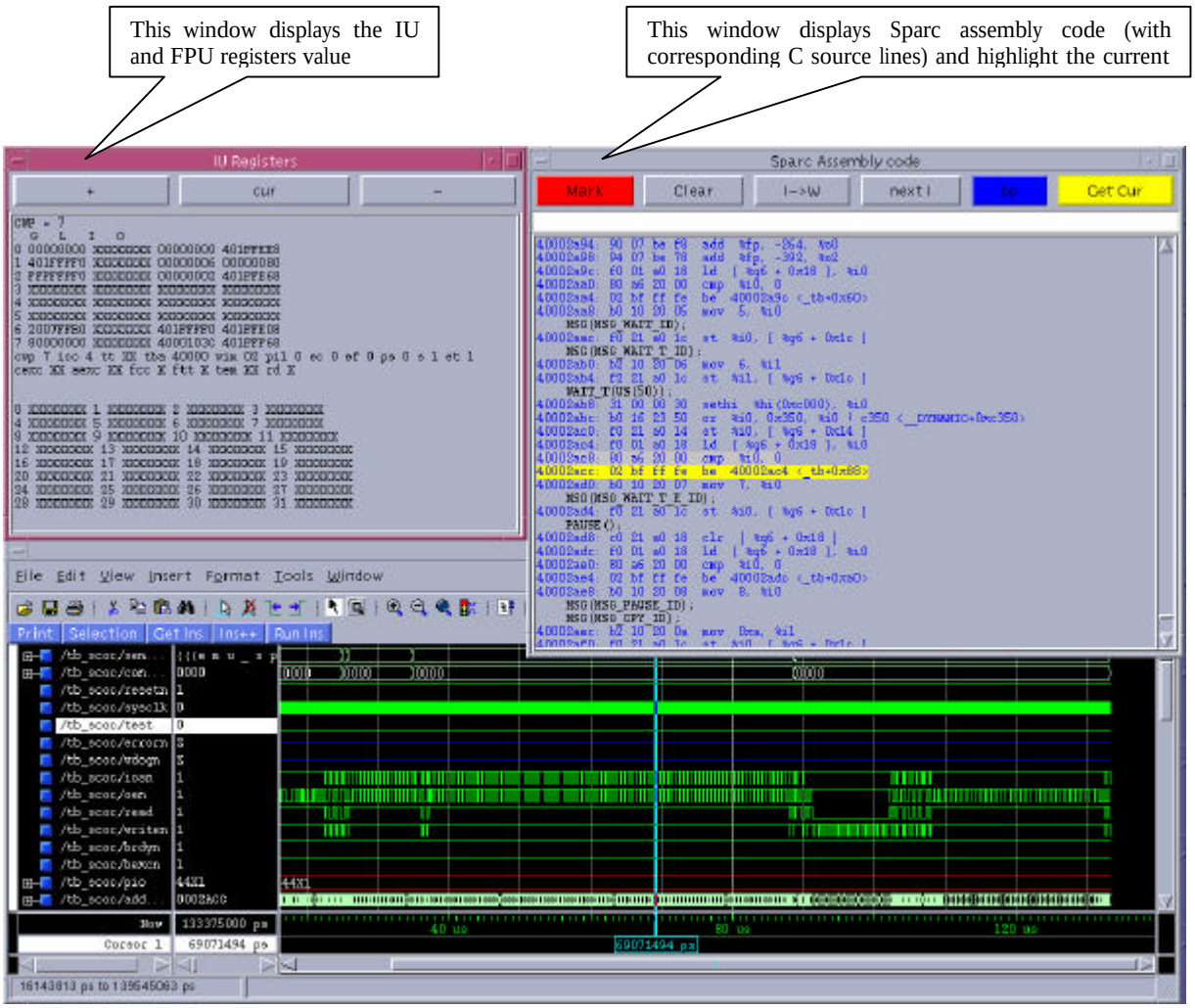

Figure 31 : Overview of the Modelsim simulation

The Modelsim scripting capabilities allow highlighting the line of code currently executed at position of the cursor in the wave window, easing the debugging of hardware/software interactions.

7.3 SYNTHESIS TOOL

Synplify is a powerful synthesis tool for quick FPGA development. The TCL scripting possibilities ease the use of this tool for complex SoC designs.

Modular or global synthesis

Synthesis can be done block by block and then the obtained netlists assembled as black boxes at top level. Black box assembly has been done for the PCI core. The main advantage of this solution is that the re-synthesis of the SoC after a modification is executed faster, as only the modified part of the design needs to be reworked. The drawback of this solution is the limitation of trans-IP optimisation.

As the SCoC FPGA was almost full (97% of the FPGA for partial configurations of SCoC), only global synthesis has been performed, allowing logic optimisation. This solution was affordable as the synthesis time was reasonable (1,5 hours).

Global synthesis with IP local scripts

One of the first tasks, and frequent source of errors, is the definition of the VHDL files describing the overall design architecture : SCoC is described by 160 VHDL files. The idea is then to define a VHDL source list locally (for each IP or library) and to re-use this source list for top level synthesis.

The other idea is also to re-use this list when changing of synthesis tool, as most of the currently available tools support TCL scripting. Also, modifications of the VHDL file architecture for an IP then only generate a modification of the local file list, without affecting global synthesis script.

Constraints

Synplify integrates a powerful constraint editor (allowing cross probing between a graphical RTL view and the constraint editor). Many constraints can be added to the design relative to timing issues, signal preserving or resource usage.

For the SCoC design, we nevertheless had to integrate Synplify attributes into the VHDL code, as the corresponding attribute was not available for insertion in a constraint file. This reduces the possibility of reuse of the design with another synthesis tool. Indeed, constraints are spread over all the IP cores and the information is not centralized into a single constraint file.

8 EVOLUTION OF STUDY

Possible evolutions of the design, of the IP and of the method have been pointed out in the different sections of this document. They start from the acquisition of detailed system requirements for performances, to the evolution of the LEON structure, the study of different bus architecture and the development of corresponding VHDL resources, the evolution of the existing IP cores. Following the first evaluation of the performances of the SCoC on the BLADE board, it is also necessary to push forward the full verification of the system, to address the testability issue (including the LEON2 evolution) prior to reach a state were we can envisage to put the system in a real ASIC.

System requirements

In this project, system requirements were established in terms of function to have, not in terms of performances. Then the first step of the evolution of the study is, based on the existing SCoC example, to establish and validate the requirements for performance of each function. This includes CPU computation performances, data throughput on the Spacewire links, data transfers on the PCI (all these performances simultaneously).

Evolution of LEON structure

LEON VHDL structure shall evolve to clearly separate the CPU IP functionality from the AMBA AHB and APB bus structure. LEON core functionalities consist in the IU/FPU/CACHE and interrupt subsystem. Other instances in LEON can be considered as a catalogue of IP : UART, TIMER, Memory Controller, etc…

Evolution of IP

Taking into account potential user feedback, it is possible to improve the IP performances, in particular for the Spacewire and the HDMA.

In addition, it is of all interest to develop a AMBA-PCI bridge available for the European space community, not based on commercial IP core.

SCoC or SCoC evolution verification

The hardware evaluation performed within this activity is not sufficient to fully verify the SCoC design. A full validation the SCoC design should be performed, along with a validation of the implemented IP cores.

9 CONCLUSION

The study performed for this contract put in evidence the absolute necessity of a reliable and performant design methodology of SoC development. It allows speeding up the architectural development phase of a large design as shown for the development of the SCoC.

This methodology is mainly based on the availability for the European space community of a library of validated IP cores. This is now well supported by ESA, as indicated by the development of the LEON-FT CPU core, the PTME, the Spacewire IP, the HurriCAN controller, the AMBA to PCI wrapper and a collection of small but indispensable IP.

The methodology is also based on the use of standards coming from commercial applications, military developments or space industry, such as AMBA busses for internal connections of the IP cores, and also external interface standards, that allows the reuse of the developed IP core, such as PCI, Mil-Std-1553, Spacewire, CCSDS TM/TC. This strategy was beneficial for the SCoC development as it allowed reusing already developed validation tools (emulators for the simulations and standard hardware testbenches for the board evaluation).

The promotion for using these standards for ESA programmes is absolutely necessary to enhance the catalogue of available IP cores and the availability of verification/validation tools.

The methodology is also based on the availability of large commercial PLD that allow rapidly prototyping the system and functionally validating it deeper than with only simulations. FPGA prototyping also provides a breadboard platform for early software development, well before the availability of the System-On-ASIC.

The development of a demonstration/evaluation hardware platform does not suppress the essential simulation phase. Nevertheless, due to the complexity of the SoC, the simulation time dramatically increases (while partially compensated by the increase of the simulation platform computation performances). The simulations are then used to fully verify the IP core functionality in a standalone mode, while the system simulations only partially verify the core functionality of each IP, focusing on the correct integration of the SoC.

The main conclusions of the SCoC activity are then :

- The development of SoC including the CPU core such as the designed SCoC is crucial for the development of future small space platforms and probes
- There is a great interest of the SoC methodology for the future development of highly integrated ASIC, including or not the CPU core.
- The methodology of IP reuse shall be promoted by the Agency in order to complete the already available catalogue of IP cores, and to enhance the verification level of these IPs.
- The software tools already used for the ASIC development (for VHDL simulation and synthesis) are still suitable to the development of SoC, provided proper scripting is used.

- The hardware prototyping of the SoC using large commercial PLD greatly improves the system verification, compensating for the reduction of verification coverage of post layout simulations.

For the study continuation, potential user's of the system shall validate the requirement and functional specification of the developed SCoC. Then a full verification of the functionalities and performances of the current or enhanced design can be performed. Finally, manufacturing a prototype ASIC and its integration in a complete Data Handling System evaluation platform will validate all the concepts of these developments, from the interest in the product to the global methodology.

---%---%---%---%---