Extending a modern RISC-V vector accelerator with direct access to the memory hierarchy through AMBA 5 CHI

Final Document

by

Miquel Roset Julia

Director: Alireza Monemi (BSC)
Co-Director: Nehir Sonmez (BSC)
Ponent: Miquel Moretó (UPC)
GEP Tutor: Joan Sarda (UPC)

Computer engineering specialization
UNIVERSITAT POLITÈCNICA DE CATALUNYA

January 2022

Abstract

The BSC is developing a decoupled RISC-V-based vector accelerator. In the previous version of this project, the vector accelerator uses the Open Vector Interface (OVI) to access the shared L2 cache, through a scalar processor core. Furthermore, the processor core accessed the shared L2 cache via the NoC. This two-level memory access mechanism introduces a significant latency overhead to the system. Moreover, memory access time is critical to the performance of the accelerator. To attack this problem, this project designs an AMBA 5 CHI interface IP that provides the accelerator with direct access to the NoC, thus reducing memory latency.

This project aims to obtain a functional design with basic operations to facilitate the integration phase. Therefore, the interface IP has no specific area or power constraints. The IP design covers different AMBA 5 CHI protocol aspects, assembles the architecture, and proposes a set of tests to verify the functionality of the designed module. Final results show that this IP can successfully provide the accelerator with direct access to the shared L2 cache, replacing the OVI interface and improving performance. We also give pointers on how to further improve the AMBA 5 CHI interface IP in terms of performance and functionalities.

Keywords: RISC-V, Network on Chip, AMBA 5 CHI, Level 2 cache, vector accelerator

Resumen

El BSC está desarrollando un acelerador vectorial desacoplado basado en RISC-V. En la versión anterior de este proyecto, el acelerador utiliza Open Vector Interface (OVI) para acceder a la caché L2 compartida, a través del núcleo escalar del procesador. Posteriormente, el núcleo del procesador accede a la caché L2 compartida a través del NoC. Este mecanismo de acceso a la memoria en dos niveles introduce un aumento considerable en la latencia de los acceso. Además, el tiempo de acceso a la memoria es fundamental para el rendimiento del acelerador. Para atacar este problema, este proyecto diseña una IP que hace de interfaz para AMBA 5 CHI y que proporciona al acelerador acceso directo al NoC, reduciendo así la latencia de los accesos a memoria.

Este proyecto pretende obtener un diseño funcional con operaciones básicas para facilitar la fase de integración. Por lo tanto, el diseño no tiene restricciones ni de área, ni de energía. El diseño cubre diferentes aspectos del protocolo AMBA 5 CHI, ensambla la arquitectura y propone un conjunto de pruebas para verificar la funcionalidad del módulo diseñado. Los resultados finales muestran que esta IP puede proporcionar con éxito al acelerador acceso directo al caché L2 compartida, reemplazando la interfaz OVI y mejorando el rendimiento. También brindamos sugerencias sobre cómo mejorar aún más la interfaz IP AMBA 5 CHI en términos de rendimiento y funcionalidades.

Palabras clave: RISC-V, Network on Chip, AMBA 5 CHI, Level 2 cache, acelerador vectorial

Resum

El BSC està desenvolupant un accelerador vectorial desacoblat basat en RISC-V. A la versió anterior d'aquest projecte, l'accelerador utilitza Open Vector Interface (OVI) per accedir a la memòria cache L2 compartida, a través del nucli escalar del processador. Posteriorment, el nucli del processador accedeix a la memòria cache L2 compartida a través del NoC. Aquest mecanisme d'accés a la memòria en dos nivells introdueix un augment considerable en la latència dels accessos. A més, el temps d'accés a la memòria és fonamental per al rendiment de l'accelerador. Per atacar aquest problema, aquest projecte dissenya una IP que fa d'interfície per a AMBA 5 CHI i que proporciona a l'accelerador accés directe al NoC, reduint així la latència dels accessos a memòria.

Aquest projecte pretén obtenir un disseny funcional amb operacions bàsiques per facilitar la fase d'integració. Per tant, el disseny no té restriccions ni d'àrea, ni d'energia. El disseny cobreix diferents aspectes del protocol AMBA 5 CHI, construeix l'arquitectura i proposa un conjunt de proves per verificar la funcionalitat del mòdul dissenyat. Els resultats finals mostren que aquesta IP pot proporcionar amb èxit a l'accelerador accés directe a la memòria cache L2 compartida, reemplaçant la interfície OVI i millorant el rendiment. També brindem suggeriments sobre com millorar encara més la interfície IP AMBA 5 CHI en termes de rendiment i funcionalitats.

Paraules clau: RISC-V, Network on Chip, AMBA 5 CHI, Level 2 cache, accelerador vectorial

Table of Contents

List of Figures

List of Tables

1 Contextualization and Scope of the project

The work described in this document is conceived within a complex context. This context spans across a diversity of fields that go far beyond the technology discussed here. This becomes clear when the development of these technologies, which are the core of exascale machines, are being funded from European initiatives defined as *"an important step of a strategic plan to develop an independent and innovative European supercomputing and data ecosystem"* [3].

Therefore, this chapter aims to briefly describe the agents, both economically and/or scientifically, that take part in this ecosystem, as well as to define the base concepts from which this thesis develops.

1.1 Context

Dating back to 23 March 2017, the EU launched the EuroHPC declaration. With it, the ambition of a world's top *High Performance Computing* (HPC) infrastructure until 2022-2023 was set [4].

Since then, multiple public-private partnerships appeared, such as the *EuroHPC Joint Undertaking* (JU), created a year later, working with budgets arround 1.1 billion Euros for 2018-20, and 7 billion Euros for 2021-27 [5]. From the EuroHPC JU emerges the *European Processor Initiative* (EPI) as a broader strategy to develop an independent European HPC industry. As of September 2021, the EPI gathers 28 partners from 10 European countries [6].

Meanwhile, problematic themes like the EU's dependencies with Asia on semiconductors have increased. The European Commissioner for Internal Market since 2019 Thierry Breton states, *"While global demand has exploded, Europe's share across the entire value*

chain, from design to manufacturing capacity has shrunk. We depend on state-of-the-art chips manufactured in Asia. So this is not just a matter of our competitiveness. This is also a matter of tech sovereignty" [7].

In this race for EU tech sovereignty, the *Barcelona Supercomputing Center* (BSC) plays an important role as a research center, embracing multiple of the EU initiatives including the already mentioned EPI.

1.1.1 eProcessor

The main goal of this EuroHPC project is to build a new open source OoO (*Out of Order*) processor and to deliver the first completely open source European full-stack ecosystem based on this new RISC-V CPU. The current thesis takes place in this project within the co-design and development of a vector processor based on the RISC-V instruction set architecture [8].

This thesis implements a memory interface for a RISC-V decoupled vector accelerator. The main RISC-V OoO processor issues the vector instructions, and the accelerator deals with processing them at their entirety. For vector memory instructions, the accelerator performs accesses directly to a shared L2 cache after translating virtual addresses with a private TLB. It also ensures that memory ordering is maintained and uses/updates all related control and status registers (CSRs).

1.2 Terms and concepts

1.2.1 European Processor initiative

European initiative focused in the following three objectives.

- Develop low-power processor technology to be included in a European pre-exascale (capable of around 10^{16} calculations per second) and exascale systems (10^{18} calculations per second).
- Guarantee that a significant part of that technology is European and
- Ensure that the application areas of the technology are not limited only to HPC, but cover other areas such as the automotive sector or the data centres [3].

1.2.2 RISC-V

Risc-V is an open standard *instruction set architecture* (ISA). Introduced in 2010, it is based on the classic RISC architecture from UC Berkeley. Aside from its notable features, *RISC-V* has experimented a rise in popularity since it allows smaller device manufacturers to build hardware without paying royalties and allow developers and researchers to design and experiment with a proven and freely available instruction set architecture (ISA) [9].

1.2.3 Vector Architecture

Vector architectures group sets of data elements scattered about memory, place them into large, sequential register files, operate on data in those register files, and then disperse the results back into memory. This means that a single instruction operates on vectors of data resulting in dozens of register-register operations on independent data elements [10, p. 264].

1.2.4 Vector Processing Unit

A Vector Processing Unit is the unit of a SoC (*System on Chip*) which performs the actual work of computing the vector instructions that the ISA in use specifies.

1.2.5 Network on Chip

This type of network is used for interconnecting microarchitecture functional units, register files, caches, compute tiles, and processor and IP cores within chips or multichip modules [10, Appendix F, p. 3].

1.2.6 Memory hierarchy

Memory hierarchy is an economical solution for supporting access to large amounts of fast memory. It takes advantage of locality and trade-offs in the cost-performance of memory technologies. Since fast memory is expensive, a memory hierarchy is organized into several levels – each smaller, faster and more expensive per byte than the next lower level, which is farther away in terms of access latency, from the processor [10, p. 72].

1.2.7 AMBA 5 CHI

AMBA 5 CHI is the latest generation of freely-available AMBA protocol specifications. It introduces the *Coherent Hub Interface* (CHI) architecture, which defines the interfaces for the connection of fully coherent processors and high-performance interconnects [11].

1.2.7.1 Request Node

A Request Node, also referred as RN, generates and sends AMBA 5 CHI transactions, including reads and writes. These requests are sent to the interconnect were other AMBA 5 CHI elements, such as Home Nodes (HN) generate the corresponding responses.

1.3 Motivation

As stated earlier in this chapter, the eProcessor architecture is composed of three main components. These three components are the core, the interconnection network or NoC and multiple domain specific accelerators, such as the vector processor and the AI and Bioinformatics accelerator.

Figure 1.1 shows the layout of the top main components of the processor. Additionally we can observe how the TLBs and the caches complete the design. The scope of this thesis is to link some of the accelerators to the NoC from where the Shared L2 cache is accessible. Therefore, this document won't elaborate on the rest of the design.

1.3.1 Accelerators

eProcessor will have a vector processing unit (VPU) intended to improve the performance of High Performance Computing (HPC) and Bioinformatics applications and two distinct accelerators for Artificial Intelligence (AI) applications: one on-chip unit and another one off-chip.

As of the iteration previous to this thesis, the memory interface that the accelerators utilized to made his way to memory worked through the core of the system. This was accomplished with the implementation of an open source spec, supported by both, the processor core in use, and the vector accelerator, called *Open Vector Interface* (OVI) [12]. Moreover, the processor core used the AMBA 5 CHI protocol to access the second level of the memory hierarchy. This way to memory had a performance issue even though OVI is specially designed for the core to accommodate functional units like vector accelerators.

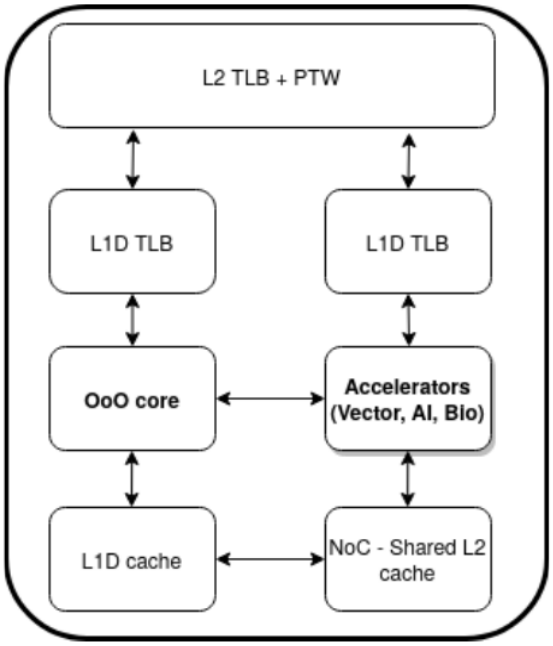


Figure 1.1: OoO core, on-chip accelerators, caches and TLBs

The overhead causing a performance degradation happened because the VPU requests need to go through the processor core first and AMBA 5 CHI afterwards.

One of the main characteristics that will be newly designed in eProcessor is to provide direct access to the memory hierarchy for on-chip accelerators. This will improve the performance of memory accesses while enforcing simple designs. This feature requires that the accelerators support the CHI interface, implementing a vector Load/Store Unit (VLSU) and a private L1D vector TLB that accesses a L2 TLB.

1.4 Stakeholders

This section specifies the agents who will benefit from the work done in this thesis.

1.4.1 BSC

The closest beneficiary from this work will be the BSC. The improvements developed in this work are planned to be included in the project's next iteration.

1.4.2 Scientific community

Because of the ISA chosen, RISC-V, the project as a whole work towards an open-source ecosystem. This ecosystem presents advantages in multitude of aspects. For example, the cost of software can dramatically reduce by enabling more reuse. This way, communities such as the scientific one, can break free from expensive ISA's which limit their capabilities. Another example would be hardware providers who, without a proprietary ISA, currently have close to none opportunities to enter the market.

1.5 Justification

The problem to be solved in this thesis is conceived as an improvement to an already working product. The problems identified in previous sections face two situations. From one side, a lack of quality, as for the performance degradation, and an unreached objective regarding to the dependency between the vector accelerator and the core.

The dependency problem is tackled by removing as much interaction as possible between the VPU and the core. In our situation, OVI represents an issue. Regarding the overhead of the OVI and AMBA 5 CHI interfaces, the idea is to connect the VPU directly to the memory hierarchy thus getting rid of the OVI interface.

An important thing to remark is that the dependency problem is the most important from the two and that there are no other solutions a part from the one that has been commented. This means that even though the performance in the final result is not as expected, it will mark the starting point for improvements since it will satisfy the independence wanted for the VPU.

1.6 Project scope

The scope of this thesis is limited to the propagation of the operations issued by the VLSU to the NoC. It's important to remark that the NoC uses the AMBA 5 CHI protocol.

Within this scope, the following global objectives, sub-objectives, functional requirements, obstacles and risks emerge.

1.6.1 Objectives

- **Understand AMBA 5 CHI:**

  - Comprehensive reading of the protocol and concepts in use.
  - Use an implementation of the protocol provided by BSC to simulate specific situations of the protocol.

- **Understand the starting state of the vector accelerator:**

  - Locate the exact place from which the VPU will connect to the memory hierarchy.
  - Understand and communicate with teams in charge of the modules surrounding that point.
  - Agree on an interface with the team members issuing memory operations which is both efficient and effective for its purpose.

  BSC already has a working model that implements the AMBA 5 CHI protocol but it has plenty of functionalities that the VPU does not require.

- **Adapt a working model, provided by BSC, to the needs:**

  - Understand the basic organization of the working model that implements the protocol.
  - Design a module which adds what is missing from the working model and eliminates what is unneeded.

  At this point, there is a synchronization with the teams that develop the modules from the VPU that interact with our module. Once their circuit is implemented the implementation of the module designed can start.

- **Implement the design:**

  - Implement the design.
  - Test and verify the results.

1.6.2 Requirements

Next, we'll enumerate and briefly explain what types of requirements this project has.

- Meet the requirements specified by the team issuing memory operations.

- Meet the guidelines and criteria established by the VPU developers team including tests and documentation.
- Make a hardware efficient design.
- Test and verify all implemented functionalities.

1.6.3 Obstacles and risks

During the development of the project multiples obstacles and risks can appear. Here is a short list of some of them.

- **Synchronization delays:** At one point in the project, the material needed is provided by another team. In this case, the deliver might get delayed and it is needed some mechanisms to reduce its impact on the development of the project.
- **Implementation errors:** Good development practices reduce the time spent fixing implementation errors. Despite this, an unnoticed mistake can take weeks, even months to get fixed.
- **Design errors:** This type of errors are the ones with greater damage potential. Since they are at the beginning of the development process, a mistake here can lead to restarting regardless of the phase you are on.

1.7 Methodology and rigour

1.7.1 Task management

Hardware development is usually more complex than software development. Famous methodologies such as Agile are unfeasible in this field and it usually falls back to the classic Waterfall. Besides that, a *Kanban* methodology is used to organize the tasks.

This methodology is very comfortable since it's widely supported by nowadays commercial tools such as *Trello*. It allows you to visually group tasks and move forward as you complete them. It also has plenty of features to comment, tweak and share the work.

1.7.2 Shared Google Drive

This tool has been chosen since it offers a large set of tools going from spreadsheets to design tools to draw block diagrams. It also makes the sharing of the work straight forward and, since it is on the cloud, there is no possible data loss.

1.7.3 Code repository

The tool to maintain and develop the code base is Git. This distributed version control system is widely embraced by the developer community. The project team at BSC enforces it and the platform in use is *GitLab*.

The module developed in this thesis is derived from a repository containing full implementations of the AMBA 5 CHI protocol. This project develops a simpler version of one of the components and joins the final design to the actual VPU repository.

1.7.4 Meetings

In order to coordinate this work, weekly one hour meetings have been organized. One slot for all the team members developing the VPU and another slot to discuss the AMBA 5 CHI more in detail as well as coordinate with the people in charge of the nearby modules.

2 Time planning

This chapter aims to define and schedule tasks involved in completing the project. The time frame dedicated to the project begins on August 1, 2021 and ends in late January 2022. During this period, 7 hours a day will be devoted.

In order to further work in the time planning, this chapter includes a Gantt diagram representing the tasks scheduling and dependencies.

2.1 Project Management

This section enumerates and defines those tasks involved in the project management.

- **PM.1 - Context and Scope:** Contextualize the thesis and describe the scope.
- **PM.2 - Time Planning:** Enumerate and describe tasks, elavorate estimates and develop a risk management strategy.
- **PM.3 - Budget and sustainability:** Elaboration of a budget and a sustainability report.
- **PM.4 - Project Documentation:** Complete and compile generated documentation in a single document together with support material. Roughly 18 hours of dedication after the completion of PM.1, PM.2 and PM.3.
  - **PM4.1:** After the completion of [VFY.1], the documentation task should continue until the delivery date.
- **PM.5 - Meetings:** Two weekly one-hour meetings. One for coordination with the working group and another one to coordinate with the project.

2.2 Project Development

This section enumerates and defines those tasks involved in the completion of the project from the technical side.

2.2.1 Familiarize with RN-I from AMBA 5 CHI

Check how the IO Coherent Request Node (RN-I) works. With this, we understand the VPU entry point to the NoC, which uses AMBA 5 CHI protocol.

- **[CHI.1] - Understand the protocol:** Using the specification provided by ARM, read and understand the insights of the protocol.
- **[CHI.2] - Understand an RN-F module:** With an implementation of an RN-F provided by BSC, read, compile and understand the project. Depends on [CHI.1].
- **[CHI.3] - Simulate CHI transactions** With the provided working model, simulate some simple transactions and contrast them with the protocol specification. Depends on [CHI.2].

2.2.2 Familiarize with the Vector Processor Unit

Check how the VPU is implemented.

- **[VPU.1] - Read about VPU architecture:** Find memory model in use, interfaces with the core, ISA supported, etc. Depends on [CHI.3].
- **[VPU.2] - Place RN-I within the VPU:** Understand where the RN-I will be placed within the VPU architecture. Depends on [VPU.1].

2.2.3 Define an interface between VPU and RN-I

Coordinate with teams from the VPU in charge of memory operations to define an interface that satisfies requirements from both sides.

- **[IF.1] - Contrast requirements:** Understand requirements imposed from the VPU teams. See whether they can be satisfied from an RN-I node defined in AMBA 5 CHI issue C [2]. Depends on [VPU.2].

- **[IF.2] - Agree on an interface:** Once [IF.1] is positive, commit to an interface which is both efficient and effective.

2.2.4 Design a model for the RN-I

Design the working model for the RN-I.

- **[DSGN.1] - Remove unnecessary logic:** BSC provides a model and implementation of a RN-F. Eliminate the logic that implements functionalities without support from RN-I.

- **[DSGN.2] - Add support for the interface:** Adapt the resulting circuit from [DSGN.1] to support the defined interface with the VPU.

2.2.5 Implement the design of the RN-I

Tasks involved with the implementation of the design. These tasks are performed iteratively over parts of the overall design. This ensures higher confidence on code over long periods of implementation.

- **[IMP.1] - Code the design:** Specify the design and the necessary verification code with System Verilog.

- **[IMP.2] - Verification code:** Write verification code for the design implemented. Depends on [IMP.1].

- **[IMP.3] - Register Transfer Level simulation:** Run and check RTL simulations. Depends on [IMP.2].

2.2.6 Additional verification process

Perform additional tests defined by the project in order to further verify the design.

- **[VFY.1] - Gate Level Simulation:** Run and check Gate Level Simulations.

2.3 Human and material resources

All material resources [MR] and human resources [HR] are used at any given task. The resources used in this project are enumerated identified below.

- **[MR.1] - Electronic devices:** The personal computer as well as the infrastructure deployed in BSC servers.
- **[MR.2] - Work place:** Office material, essentials, etc.
- **[HR.1] - Team of engineers:** Superiors, coworkers, etc.
- **[HR.2] - Other roles**: Employees from human resources, maintenance and other departments required.

2.4 Estimates and Gantt representation

In this section we complete the task description with estimates of the tasks duration as well as the dependencies between them. This is done through a Gantt representation.

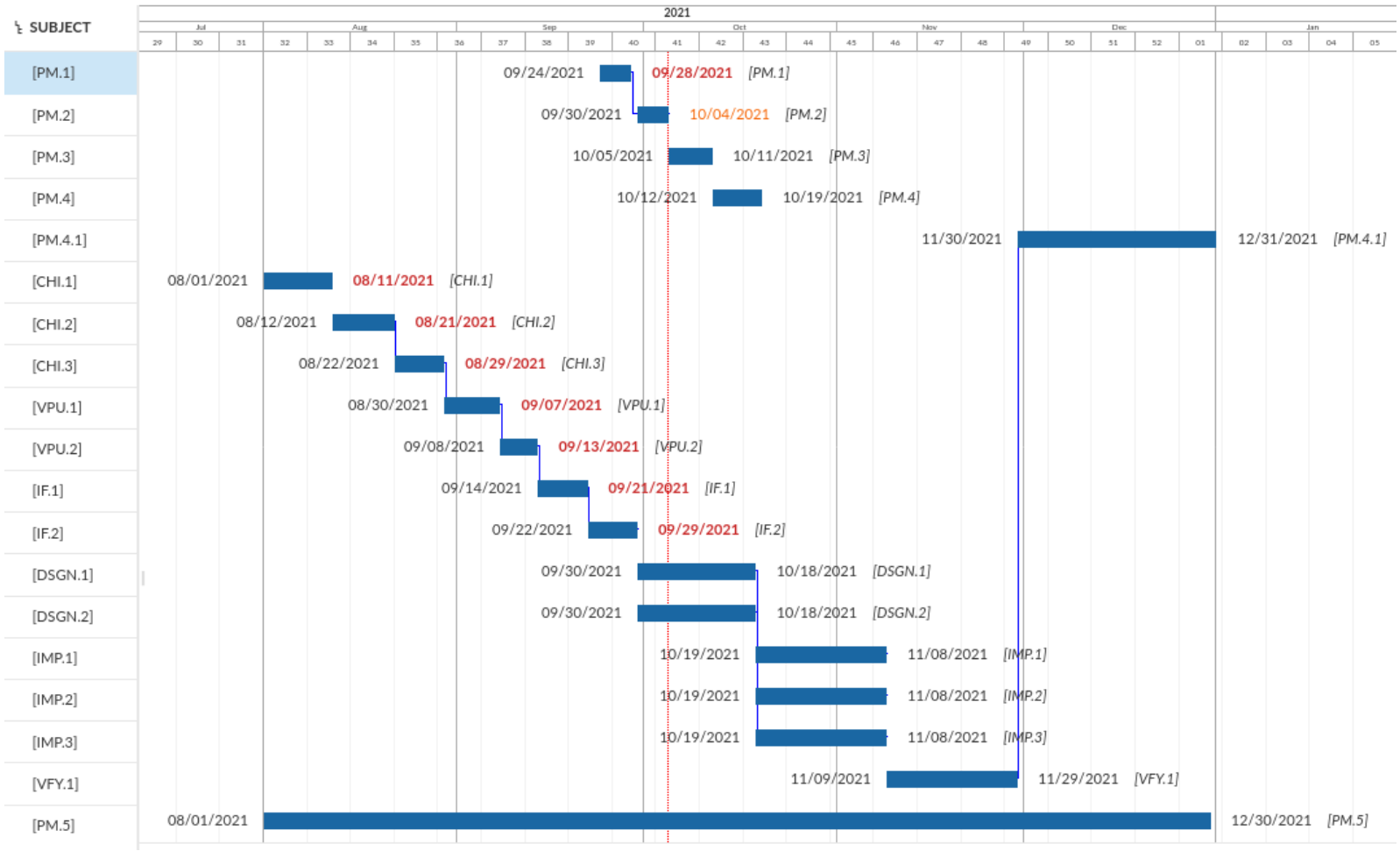


Figure 2.1: Gantt diagram showing duration and dependencies of tasks.

2.5 Tasks summary

This section summarizes the tasks enumerated and described through this chapter.

Table 2.1: Task summary

ID	Name	Duration (h)	Predecessors	Resources
PM.1	Context and Scope	18	-	HR, MR
PM.2	Time Planning	18	PM.1	HR, MR
PM.3	Budget and sustainability	18	PM.2	HR, MR
PM.4.1	Project Documentation	18	PM.3	HR, MR
PM.4.2	Project Documentation	18	VFY.1	HR, MR
PM.5	Meetings	52	-	HR, MR
CHI.1	Understand the protocol	70	-	HR, MR
CHI.2	Understand an RN-F module	42	CHI.1	HR, MR
CHI.3	Simulate CHI transactions	35	CHI.2	HR, MR
VPU.1	Read about VPU architecture	49	CHI.3	HR, MR
VPU.2	Place RN-I within the VPU	28	VPU.1	HR, MR
IF.1	Contrast requirements	42	VPU.2	HR, MR
IF.2	Agree on an interface	42	IF.1	HR, MR
DSGN.1	Remove unnecessary logic	91	IF.2	HR, MR
DSGN.2	Add support for the interface	91	DSGN.1	HR, MR
IMP.1	Code the design	105	DSGN.2	HR, MR
IMP.2	Verification code	105	IMP.1	HR, MR
IMP.3	Register Transfer Level simulation	105	IMP.2	HR, MR
VFY.1	Gate Level Simulation	105	IMP.3	HR, MR

2.6 Risk management: alternative plans and obstacles

This section describes the management of the risks and obstacles defined previously. It also describes alternative tasks, any additional resources that might be needed and the overall impact over the project.

In case of encountering an obstacle during the development of [IMP.1], [IMP.2] and [IMP.3], the cyclic alternation of them allows to start over and keep progressing. This means that there is no need for alternative tasks to deal with [IMP] obstacles.

Both [DSGN.1] and [DSGN.2] have huge potential in causing delay. In case of encountering an obstacle for this tasks, which could happen time after completing them, the alternative tasks can be from the groups of [CHI] and [VPU] if they are due to misunderstanding of those subjects and from [DSGN] if redesign is needed. They can also be from [IMP] task group if the implementation can make up for the design obstacle.

Regarding the [VFY] task, obstacles encountered here usually require minor changes from previously completed tasks.

None of the obstacles encountered need additional material resources. If obstacles encountered cannot be solved autonomously, additional human resources might be needed in form of specific help from coworkers.

Finally, the maximum delay that this project can cope with is one month. Since the deadline to deliver the project is in late January and the Gantt diagram shows the completion of all tasks by the end of December, there is a hole month to extend delayed tasks, if ever needed.

2.7 Changes in the planning

Regarding the planning exposed, there has been just one major change. This change affects the Gate Level Simulation (GLS) or [VFY.1] task in the GEP document. Due to the nature of the test, it should be carried out as a post-layout netlist verification once almost all the technical parameters are known. Since at the time planned to carry out this task the design of the VPU is incomplete, most of the parameters to carry out a GLS remain indeterminate. In the place of [VFY.1], [IMP.3] has been extended on time and the modules targeted for the RTL simulations have also been enlarged to the whole NoC including different models of Home Nodes.

3 Budget

3.1 Identification of Costs

To perform the cost identification we start by identifying the different types of direct costs involved in the achievement of the project.

- **[HR.1] Team of engineers:** Workers involved in the project. The role that performs most of the tasks of this project is a Undergraduate Student [UE]. The rest of the roles involved will be considered an average engineer [ENG] with an average salary [13]. Despite this, actual roles might be:

  - Master's Student
  - PhD Student
  - Recognized Researcher
  - Established Researcher

  The team involved in the tasks is composed by two Undergraduate Students and 6 Engineers. All of them working full-time except from one Undergraduate Student.

  Regarding the distribution of the tasks from this project among the roles described above, the two Undergraduate Students will participate in all the [CHI], [VPU], [IF], [DSGN], [IMP] and [VFY] task groups. Additionally, three more Engineers will take part in [IF]. A single Undergraduate Student will perform the [PM.1], [PM.2], [PM.3] and [PM.4] tasks. Finally, regarding [PM.5], one meeting will be attended by two Engineers and two Undergraduate Students, and the second one will be attended by two Undergraduate Students and 6 Engineers.

Another type of costs that this project includes come from the material resources.

- **[MR.1] - Electronic devices:** The most significant elements from this are software licenses from *Questasim* 2019.4 [14] and the computational use time of 3 servers we were given access to.

- **[MR.2] - Work-place:** Multiple costs derive from the work place at the office. These are electricity, furniture and additional infrastructures such as Internet, etc.

Table 3.1 summarizes the costs described in this chapter. It groups costs per activity (CPA), it also shows general costs (CG), Contingencies and Incidents Costs.

Table 3.1: Costs per activity (CPA)

Activity	Import (€)	Comments
PM.1 PM.2 PM.3 PM.4.1 PM.4.2	630	90 h x (1[US] x 7€/h)
PM.5	3099,2	26 h x (2[US] x 7 €/h + 2[ENG] x 11 €/h) + 26h x (2[US] + 6[ENG])
CHI.1 CHI.2 CHI.3 VPU.1 VPU.2	3.136	224 h x 2[US] x 7 €/h
IF.1 IF.2	3092,88	84 h x (2[US] x 7 €/h + 2[ENG] x 11,41 €/hora)
DSGN.1 DSGN.2 IMP.1 IMP.2 IMP.3 VFY.1	4214	301 h x (2[US] x 7 €/h)
Total CPA	**14.174,16**	Total costs per activity
Laptop	670	**Hardware** costs of desktop and EPI servers use.
2 Displays	452.54	
Keyboard & mouse	22	
Docking station	30	

EPI servers	1.000	
2 Modelsim licence	1995.01	**Software** used for simulations.
Electricity	450	
Furniture	150	
Internet Access	227	Additional needs for the **space** at workplace.
Total GC	**6.991,00**	Total of general costs.
Total CPA + Total GC	**21.165,16**	Total costs per activity plus general costs.
Total C	**3.174,77**	Total Contingency with a margin of 15 %
Total CPA + GC + C	**24.339,93**	Total CPA + GC + C
1 week Delay [IMP] 1 week Delay [VFY]	560	Cost: 40 h x 2[US] x 7 €/h. Risk: 40%
Total incidentals	1.120	
TOTAL	**25.459,93**	

3.1.1 Wages

Table 3.2 shows the wages of each role including social security tax. These wages are used to calculate the Costs per Activity (CPA), Contingency and Incidents Costs.

To obtain the human resources cost for each task, we identified the cost of the roles assigned to each task. This calculation is done with the salary of each role, the number of people doing it and the amount of hours dedicated.

Note that the Undergraduate Student role has a contract of 1750 hours per year whereas the Average Engineer has a contract of 1860 hours per year. The annual gross salary of 36.500 € for an Average Engineer is obtained from *Jobted* [13].

Table 3.2: Costs per activity (CPA)

Role	**Annual Salary (€)**	**Annual Salary with SS (€)**	**Price hour (€)**
Undergraduate Student	14.000,00	18.200,00[1]	8
Average Engineer	27.418,80[2]	36.500,00	14,74

[1] Obtained by multiplying by 1.3 the annual salary

[2] Obtanied with a tool provided by *El Pais* [15]

3.2 Management Control

This section describes the mechanisms for controlling potential budget deviations. This description will be accompanied with numerical indicators used for calculations used in deviations.

The deviation of costs in human resources (HR) for each task described in the Gantt diagram (CPA) is:

*HR deviation = Cost per hour * (Estimated hours of dedication - Real hours of dedication)*

As for the general costs (GC) it is useful to have the deviation with the amortization process:

*Amortization Deviation = (Estimated usage hours - Real usage hours) * Price per hour*

Another deviation that might occur could be with the material resources (MR). Thus its formula is:

MR deviation = Estimated resources usage - Real resources usage

One of the biggest deviations could easily be the deviation of the Incidental cost. To obtain it we use the following calculation:

*Incidental cost deviation = (Estimated incidental hours - Real incidental hours) * Total incidental hours*

Lastly, to consider all the deviations as a hole we use the following calculus:

Cost Deviation = HR Deviation + Amortization Deviation + MR Deviation + Incidental cost Deviation

The information provided by these indicators help to locate where the deviations in the budget are. This allows to quickly overview how good the estimates are during the project development.

4 Sustainability

This chapter studies how the project put into production (PPP) impacts society. This PPP includes the planning, the development and the implementation of the project.

The study made in this chapter analyzes the PPP from three different points of view. These are environmental, social and economic.

4.1 Environmental impact

Have you estimated the environmental impact that the project will have? Have you considered minimizing the impact, for example, by reusing resources?

First of all, the environmental impact of the PPP compared to the rest of process that the product of this project will go through is insignificant. This is due to tasks being very different in nature. While the result of the thesis is reached through simulations, the process after it (fabrication) will directly involve mining, manufacturing, huge water consumptions, etc.

During the development of this thesis, any task which has already been done by any other coworker is reused thus minimizing computations.

Have you quantified the environmental impact of carrying out the project? What steps have you taken to reduce the impact? Have you quantified this reduction?

The environmental impact of a hardware design project is challenging to measure. We did not have the necessary tools to carry out such measurements when developing this thesis.

Suppose we consider the circuit design produced rather than designing it; some related parameters include circuit area and power consumption. These parameters have not been analyzed in this thesis. However, they will be studied and optimized in future iterations of the design.

If you did the project again, could you do it with fewer resources?

Probably yes. The main reason would be the reuse of knowledge, designs and the development environment. These resources are worked to be scalable and instructive for when the project needs to resume.

Will the project reduce the use of other resources? Overall, will the use of the project improve or worsen the ecological footprint?

One of the most common uses of HPC is to simulate and predict. In most cases, this process saves many resources. A great example could be the simulation of plane crashes. Another example is the calculation of optimal solutions. HPC improve the results obtained in this area, providing greater solutions, faster, and more accurate.

4.2 Economic impact

Have you estimated the cost of carrying out the project?

The cost of carrying out the project has been estimated. Considering the benefits that come with the development of *eProcessor*, the costs are widely justified. Being able to pursue such goals with this budgets is clearly an opportunity that *BSC-CNS* is taking advantage of.

Has the expected cost been adjusted to the final cost? Have you justified the differences? The cost has been quantified by counting the dedication hours and the resources used. The final cost has adjusted to the expected one. It has not been necessary to justify any differences.

4.3 Social impact

What do you think the realization of this project will bring you on a personal level?

Regarding the technical side, an enormous learning of the procedures and methodologies used in the design and verification of Hardware. Regarding the industry, a first glance over dynamics, interests and synergies required to push forward a project of this kind. Last, but not least, an enrichment of the soft skills due to months of teamwork.

Has the realization of this project involved significant reflections on a personal, professional or ethical level from the people who have intervened?

The most significant reflections have been about on a professional level. The development of this project has given me the possibility to experience the process of designing a

hardware component for the first time. It has been a great opportunity to put into practice the last 4 years of study and further test my curiosity and interest in the field.

Could scenarios occur that would make the project detrimental to some particular segment of the population?

Lots of segments of the population can be affected by the product. The Hardware industry makes tools. These tools can be used by many different people and with very different intentions. Industries such as the oil and military ones might have great interest in HPC, so can medical and social researchers. The manufacturing process might also be detrimental to, for example, the workers from the factories, the miners, and many more people involved in manufacturing the product if the working conditions are not correct.

5 Background

This chapter introduces some concepts from the ICN used in eProcessor that will be used during this project. It describes the theoretical origin, the architecture in use and some terminology.

5.1 AMBA

The Advanced Micro controller Bus Architecture (AMBA) bus protocols is a set of interconnect specifications from the company ARM that standardizes on chip communication mechanisms between various functional blocks (or IP) for building high performance SOC designs. The primary motivation of the AMBA protocols is to have a standard and efficient way to interconnecting these blocks with re-use across multiple designs. [1]

Starting at 1995, AMBA 1 introduced Advanced System Bus (ASB) protocols for high bandwidth interconnect and an Advanced Peripheral Bus (APB) protocol for low bandwidth peripheral interconnects. Subsequent revisions of AMBA introduced various protocols such as APB2, AHB, AXI3, AXI4, and ACE. Figure 5.1 shows the evolution of AMBA protocols.

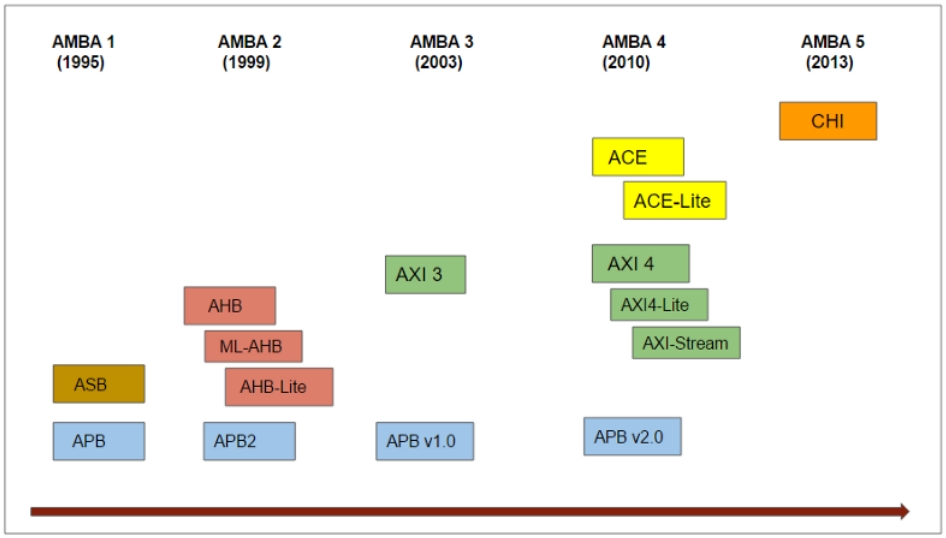


Figure 5.1: Evolution of AMBA protocols [1]

5.2 AMBA 5 CHI

The AMBA 5 revision introduced the Coherent Hub Interconnect (CHI) protocol as a complete re-design of the AXI Coherence extension (ACE) protocol. CHI uses a layered packet based communication protocol with support of Quality of Service (QoS) based flow control and retry mechanisms. [1].

Since the start of CHI, multiple issues have been published. The current section is based in the issue C.

5.2.1 Architecture overview

As mentioned earlier, CHI architecture provides a layered specification. This architecture permits flexibility on the topology of the components connections, which can be driven from the system performance, power, and area requirements [2, p. 18].

Some of the key features of the architecture are [2, p. 18]:

- Scalable architecture, enabling modular designs that scale from small to large systems.
- Independent layered approach, comprising of Protocol, Network, and Link layer, with distinct functionalities.
- Packet-based communication.

Table 5.1: Layers of the CHI architecture [2, p. 19]

Layer	Communication granularity	Primary function
Protocol	Transaction	The Protocol layer is the top-most layer in the CHI architecture. The function of the Protocol layer is to: • Generate and process requests and responses at the protocol nodes. • Define the permitted cache state transitions at the protocol nodes that include caches. • Define the transaction flows for each request type. • Manage the protocol level flow control.
Network	Packet	The function of the Network layer is to: • Packetize the protocol message. • Determine, and add to the packet, the source and target node IDs required to route the packet over the interconnect to the required destination.
Link	Flit	The function of the Link layer is to: • Provide flow control between network devices. • Manage link channels to provide deadlock free switching across the network.

- All transactions handled by an interconnect-based Home Node that co-ordinates required snoops, cache, and memory accesses.
- Both MESI and MOESI cache models with forwarding of data from any cache state.
- Error reporting and propagation across components and interconnect for system reliability and integrity.

5.2.1.1 Layers

The specification differentiate three layers:

- Protocol.
- Network.
- Link

Table 5.1 describes the primary function of each layer.

5.2.2 Terminology

Transaction A transaction carries out a single operation. Typically, a transaction either reads from memory or writes to memory [2, p. 21].

Packet The granule-of-transfer over the interconnect between endpoints. A message might be made up of one or more packets. For example, a single Data response message can be made up of 1 to 4 packets. Each packet contains routing information, such as destination ID and source ID that enables it to be routed independently over the interconnect. [2, p. 21]

Flit The smallest flow control unit. A packet can be made up of one or more flits. All the flits of a given packet follow the same path through the interconnect. For CHI, all packets consist of a single flit. [2, p. 21]

Requester A component that starts a transaction by issuing a Request message. The term Requester can be used for a component that independently initiates transactions and a component is also referred to as a master. The term Requester can also be used for an interconnect component that issues a downstream Request message independently or as a side-effect of other transactions that are occurring in the system. [2, p. 21]

Completer Any component that responds to a transaction it receives from another component. A Completer can either be an interconnect component or a component, such as a slave, that is outside of the interconnect. [2, p. 21]

Slave An agent that receives transactions and completes them appropriately. Typically, a slave is the most downstream agent in a system. A slave can also be referred to as a Completer or Endpoint. [2, p. 21]

Protocol Credit A credit, or guarantee, from a Completer that it will accept a transaction. [2, p. 22]

Link layer Credit A credit, or guarantee, that a flit will be accepted on the other side of the link. A Link layer Credit (L-Credit) can be considered to be a credit for a single hop at the Link layer. [2, p. 22]

ICN A short form of interconnect, which is the CHI transport mechanism that is used for communication between protocol nodes. An ICN might include a fabric of switches connected in a ring, mesh, crossbar, or some other topology. The ICN might include protocol nodes such as Home Node and Misc Node. The topology of the ICN is implementation defined. [2, p. 22]

RN Request Node. Generates protocol transactions, including reads and writes, to the interconnect. [2, p. 22]

HN Home Node. Node located within the interconnect that receives protocol transactions from Request Nodes, completes the required Coherency action, and returns a Response. [2, p. 22]

SN Slave Node. Node that receives a Request from a Home Node, completes the required action, and returns a Response. [2, p. 22]

IO Coherent node An RN that generates some Snoopable requests in addition to Non-snoopable requests. The Snoopable requests that an IO Coherent node generates do not result in the caching of the received data in a coherent state. Therefore, an IO Coherent node does not receive any Snoop requests. [2, p. 22]

5.2.3 Transactions

As defined in Section 5.2.2, a transaction carries out a single operation. Typically, a transaction either reads from memory or writes to memory. AMBA 5 CHI defines multiple types of them. This section goes through the subset that are used in the thesis, describing them as well as their structure.

5.2.3.1 Channels overview

Communication between nodes is channel based. Table 2-1 shows the channel naming and the channel designations at the RN and SN nodes. [2, p. 32]

Table 5.2 shows the shorthand name and the physical channel name that exists on the rn or sn component.

There are three different channels of interest, REQ, DAT, and RSP.The channel fields of interest that have the same meaning across multiple channels are:

TgtID The node ID of the component to which the message is targeted. This is used by the interconnect to determine the port to which the message is sent [2, p. 315].

SrcID The node ID of the component from which the message is sent. This is used by the interconnect to determine the port from which the message has been sent [2, p. 315].

Opcode Specifies the operation to be carried out [2, p. 318].

Table 5.2: Channel naming and designation at the RN and SN nodes [2, p. 32].

Channel	RN channel designation	SN channel designation
REQ	TXREQ. Outbound Request.	RXREQ. Inbound Request.
WDAT	TXDAT. Outbound Data. Used for write data, atomic data, snoop data, forward data.	RXDAT. Inbound Data. Used for write data, atomic data.
SRSP	TXRSP. Outbound Response. Used for Snoop Response and Completion Acknowledge.	-
CRSP	RXRSP. Inbound Response. Used for responses from the Completer.	TXRSP. Outbound Response. Used for responses from the Completer.
RDAT	RXDAT. Inbound Data. Used for read data, atomic data.	TXDAT. Outbound Data. Used for read data, atomic data.
SNP	RXSNP. Inbound Snoop Request.	-

Addr Specifies the address associated with the message [2, p. 322].

RespErr Indicates the state in the CompData sent from the Snoopee to the Requester [2, p. 331].

The fields of interest exclusive to the REQ channel are:

Endian Indicates the endianness of Data in an Atomic transaction [2, p. 325].

Size Specifies the size of the data associated with the transaction [2, p. 323].

Order Specifies the ordering requirements for a transaction [2, p. 318].

MemAttr Memory attribute associated with the transaction [2, p. 323].

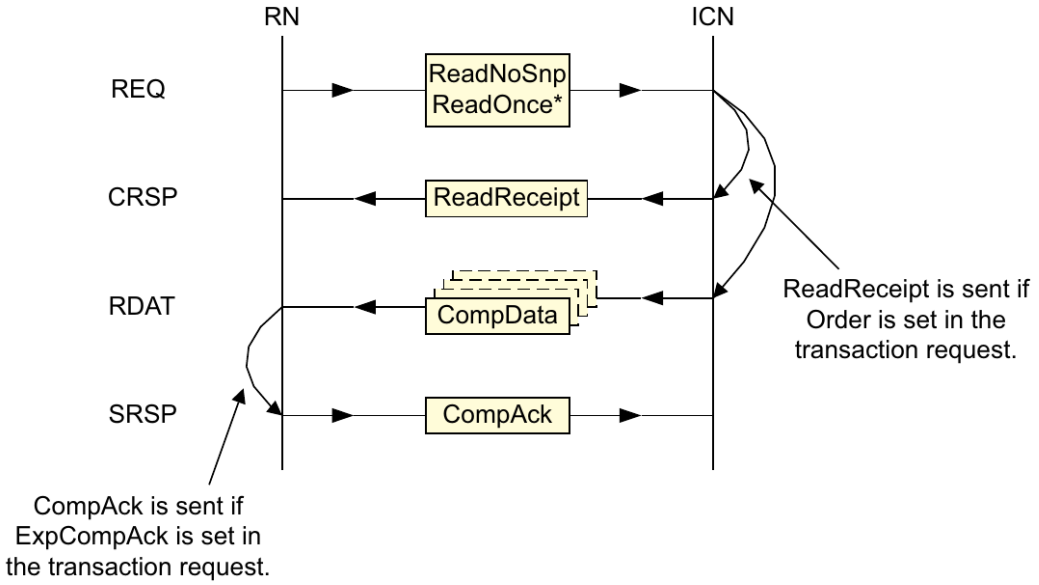
Excl Indicates that the corresponding transaction is an Exclusive type transaction [2, p. 325].

ExpCompAck Indicates that the transaction will include a CompAck response [2, p. 326].

The fields of interest exclusive to the DAT channel are:

DBID Data Buffer Identifier. The DBID field value in the response packet from a Completer is used as the TxnID for CompAck or WriteData sent from the Requester. In Snoop responses with data pull, this field value indicates the value to be used in the TxnID field of data pull response messages [2, p. 317].

Figure 5.2: ReadNoSnp and ReadOnce structure without Direct Data Transfer. [2, p. 48]



BE Byte Enable. Indicates if the byte of data corresponding to this byte enable bit is valid. [2, p. 333]

Data Data payload. This is the data payload that is being transported in a Data packet. The data bus widths supported are: 128-bit, 256-bit, and 512-bit. [2, p.332]

5.2.3.2 Transaction structure

The operations used for read transactions are *ReadNoSnp* and *ReadOnce*.

ReadOnce Read request to a Snoopable address region to obtain a snapshot of the coherent data [2, p. 143].

ReadNoSnp Read request by an RN to a Non-snoopable address region, or from HN to obtain a copy of the addressed data from the Slave [2, p. 143].

The transaction structure is the same for both, ReadOnce and ReadNoSnp. Figure 5.2 illustrates the structure when no Direct Data Transfer is used. None of the read transactions from this project include the ReadReceipt nor the CompAck messages.

The operations used for write transactions are WriteNoSnpFull, WriteNoSnpPtl, WriteUniqueFull, and WriteUniquePtl.

WriteNoSnpFull/Ptl Write a full/partial cache line of data from an RN to a Non-snoopable address region, or a write for a full cache line of data from Home to Slave [2, p. 150].

WriteUniqueFull/Ptl Write to a Snoopable address region. Write up to a cache line of data to the next-level cache or memory when the cache line is Invalid at the Requester [2, p. 150].

The transaction structure can be different between the two types of write operations. Figure 5.3 show possible structures for WriteNoSnp.

Figure 5.3: Reduced WriteNoSnp transaction structure options [2, p. 58]

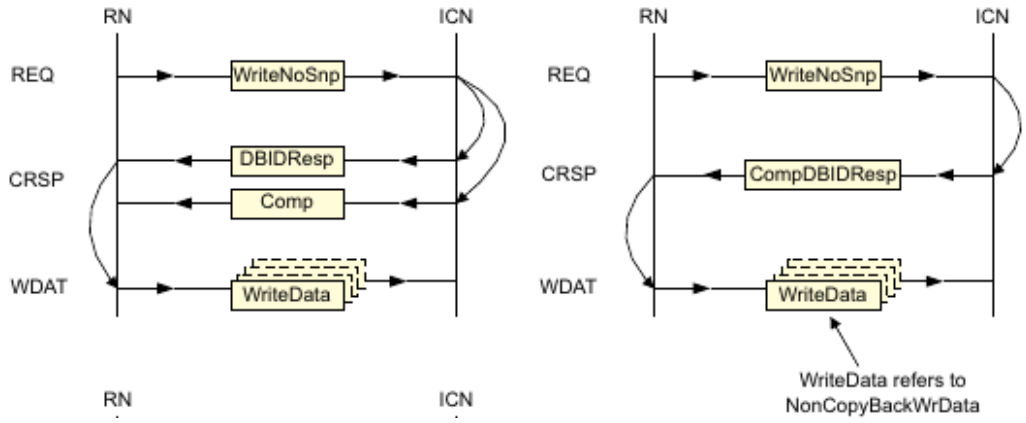


Figure 5.4 illustrates the slightly different transaction structure for WriteUnique. However, this project wont use the CompAck message. Setting the options this way, privides the same structure for both, WriteUnique and WriteNoSnp.

Figure 5.4: Reduced WriteUnique transaction structure options [2, p. 60]

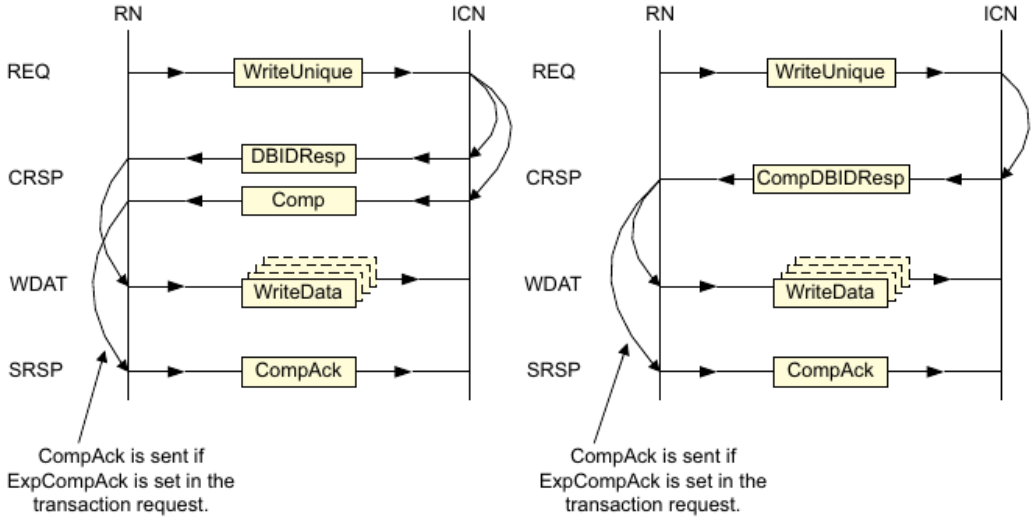
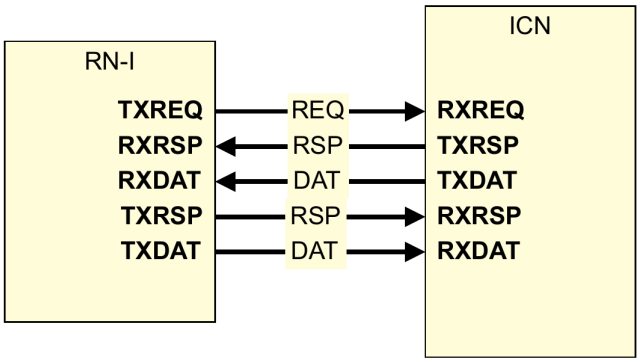


Figure 5.5: RN-I interface. [2, p. 305]



5.3 Link layer

Focusing on the node RN-I, the interface used consist of all the channels, with exception of the SNP channel. A SNP channel is not required because an RN-I node does not include a hardware-coherent cache or TLB. Figure 5.5 shows the RN-I interface. [2, 305]

Additionally, the design of the RN-I done in this project does not include the RSP channel because it is not used.

Each channel have the same set of interface pins. Figure 5.6 shows the REQ channel interface pins, where R is the width of REQFLIT. Figure 5.7 illustrates the time diagram of a flit being sent.

Figure 5.6: REQ channel interface pins [2, p. 306]

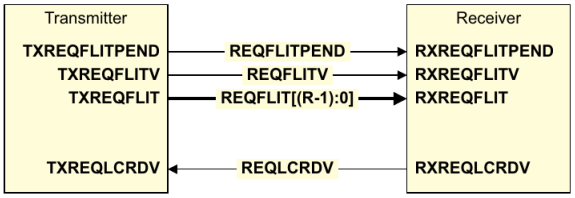
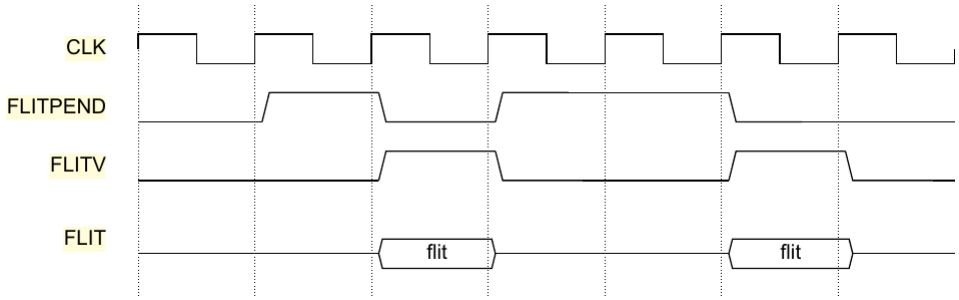


Table 5.3: REQ channel interface signals [2, p. 306]

Signal	Description
REQFLITPEND	Request Flit Pending. Early indication that a request flit might be transmitted in the following cycle. See *Flit level clock gating* on page 13-341.
REQFLITV	Request Flit Valid. The transmitter sets this signal HIGH to indicate when **REQFLIT[(R-1):0]** is valid.
REQFLIT[(R-1):0]	Request Flit. See *Request flit* on page 12-310 for a description of the request flit format.
REQLCRDV	Request L-Credit Valid. The receiver sets this signal HIGH to return a request channel L-Credit to a transmitter. See *L-Credit flow control* on page 13-339.

Information is transferred across an interface channel by the use of the *Link layer Credit* (L-Credit). To transfer one flit from the transmitter to the receiver the transmitter must have obtained an L-Credit. An L-Credit is sent from the receiver to the transmitter by asserting the appropriate LCRDV signal for a single clock cycle. There is one LCRDV signal for each channel. Each transfer of a flit from the transmitter to the receiver consumes one L-Credit. The minimum number of L-Credits that a receiver can provide is one. The maximum number of L-Credits that a receiver can provide is 15. [2, p. 339]

Figure 5.7: FLITPEND indicating a valid flit in next cycle [2, p. 341]



6 Interface proposal VPU - RN-I

This chapter describes a proposal for the interface between the VPU and the *Request Node* (RN-I). The interface provides a custom way to communicate memory operations which is far simpler than the method used to propagate them across the NoC.

Figure 6.1 illustrates the interface located in the emphasized connector between the blocks ”Load/Store Unit” and ”CHI to L2 Cache”. Notice that the set of gray blocks represents the VPU.

6.1 Modules involved

The following subsections identify the modules participating in the interface.

6.1.1 Load Store Unit (LSU)

This system is in charge of compiling needed information of memory operations and propagate them to the Request Node. It also helps to maintain memory state according to the RISC-V memory coherence model (RVWMO [16]).

6.1.2 IO coherent Request Node (RN-I)

This system is a bridge between the VPU and the NoC. It propagates the memory operations issued by the LSU by generating their corresponding AMBA 5 CHI transactions.

6.2 Transactions

This section gives an overview of the communication channels between the VPU and the RN-I, the associated packet fields, and the transaction structure.

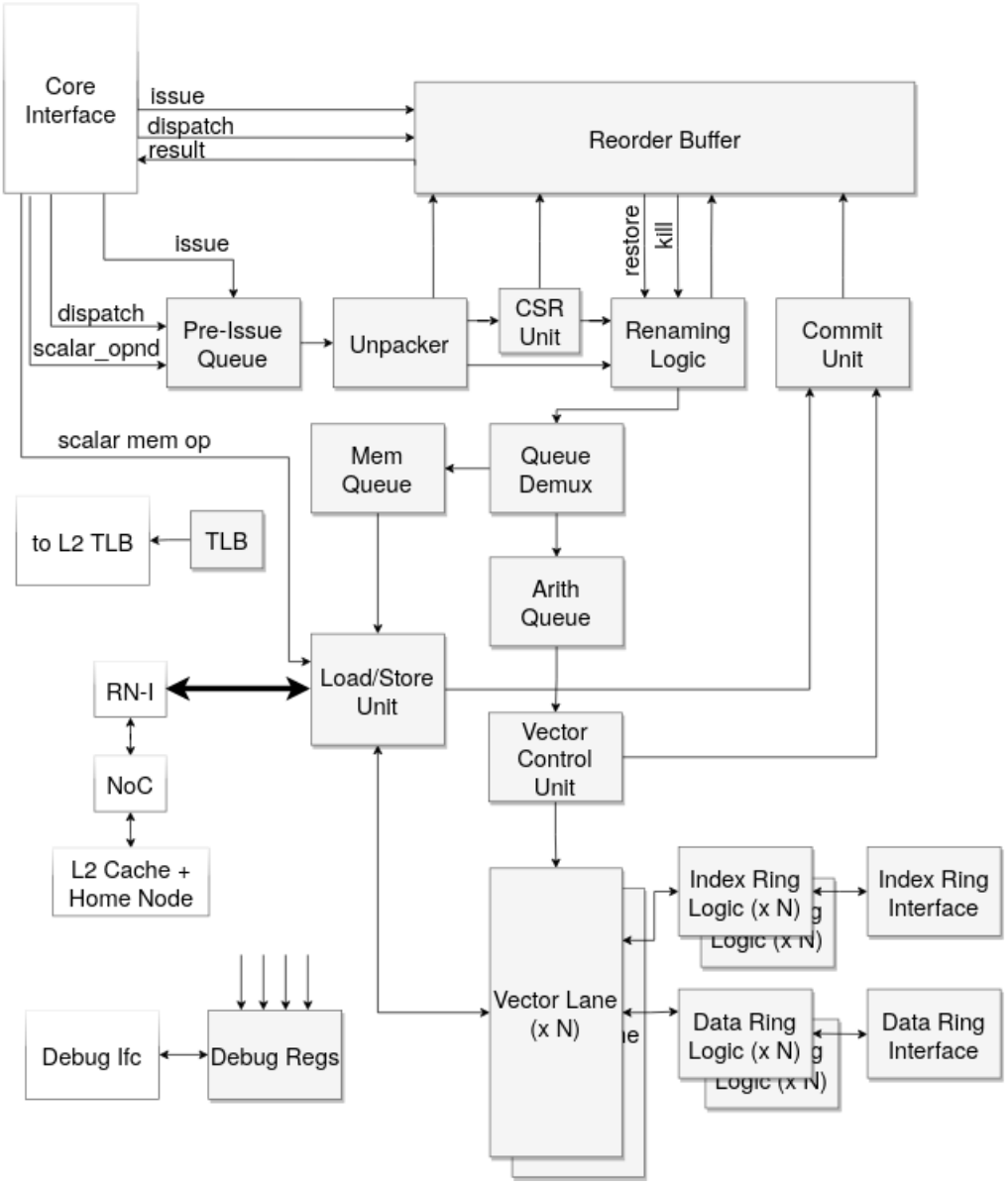


Figure 6.1: VPU block diagram

6.2.1 *Channels overview*

Communication between the VPU and the RN-I is channel based. Table 6.1 shows the channel naming and the channel designations at the LSU and RN-I.

Table 6.1: Channel's mapping and designation at the RN-I and VPU

Channel	VPU channel designation	RN-I channel designation
REQ	TXREQ. Outbound Request.	RXREQ. Inbound Request.
RDAT	RXDAT. Inbound Data. Used for read data.	TXDAT. Outbound Data. Used for read data.
RSP	RXRSP. Inbound Store Responses. Used for store operations.	TXRSP. Outbound Store Responses. Used for store operations.
WDATA	TXDATA. Outbound data. Used for write operations	RXDATA. Inbound data. Used for write operations.

6.2.2 *Channel fields and Flit packet definitions*

Table 6.2 details the payloads of the four channels previously described.

Table 6.2: Flit format and fields description

Channel	Signal	Width (b)	Description
REQ	Tag	8	Transaction identifier
	Opcode	2	Selects between Load/Store/StorePrl.
	Addr	56	Specifies the address associated with the message.
	Excl	1	Exclusive transaction type.
	Attr	1	Selects between Device/Cacheable
RDAT	Tag	8	Transaction identifier.
	Error	1	Error status.
	Data	512	Requested data
RSP	Tag	8	Transaction identifier
	Error	1	Error status.
WDATA	Kill	1	Cancel transaction.
	Tag	8	Transaction identifier
	Data	512	Data to be stored.
	Be	64	Byte Enable mask.

6.2.2.1 *Fields' constraints*

- Transactions issued through REQ channel must not have both Exclusive and Cacheable fields selected.
- The Kill flag can only be asserted for WritePtl operations.

6.2.3 *Transaction Structure*

The transactions that this interface supports have two major classifications: Read and Write. Each type of transaction has his own structure.

6.2.3.1 *Read transaction structure*

The transaction described in this subsection is used to carry out a read operation. The progress of a the Read transaction is as follows:

1. The LSU sends a Read opcode on the REQ channel.
2. The RN-I sends a Load Response with the data requested through the RDATA channel.

There is one restriction that applies to Read transactions:

- The LSU can reuse the "Tag" only after the response that uses it has been returned.

Figure 6.2 shows the transaction structure.

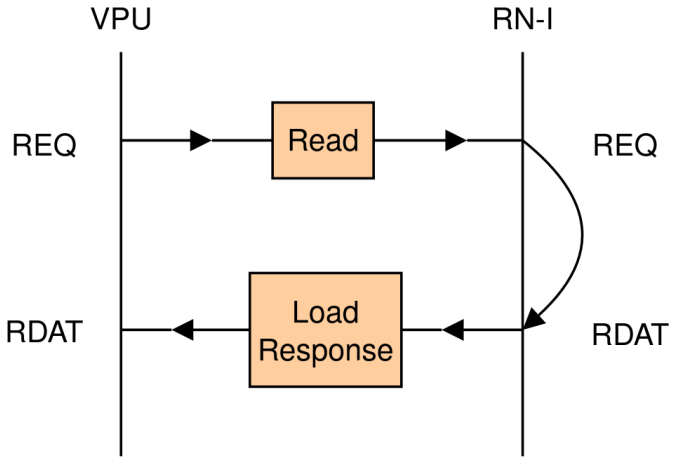


Figure 6.2: Read transaction structure.

6.2.3.2 Write transaction structure

The transaction described in this subsection is used to carry out a store operation. The progress of a the store transaction is as follows:

1. The LSU sends a WritePtl opcode on the REQ channel.
2. The RN-I returns a response to indicate that it can accept the write data for the transaction.
3. The LSU sends the write data and the associated byte enables. At this point, the LSU is able to cancel the operation through the ”kill” WDATA channel field.

The Request/Response rules are:

- Write Data must only be sent by the LSU after the Store Response.

The following restrictions apply to Write transactions:

- The LSU can reuse the ”Tag” only after the Write Data that uses it has been sent.
- Two store operations are ordered only if the second one is started after the Store Response from the first one is received.

Figure 6.3 shows the transaction structure.

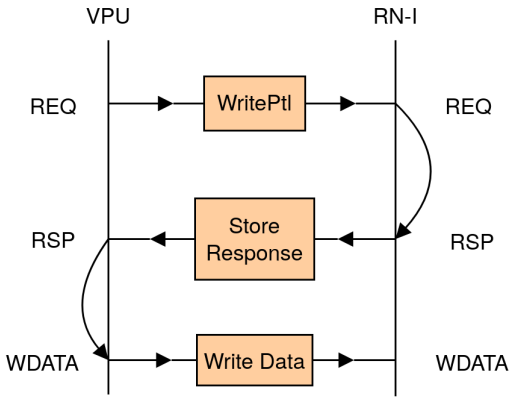


Figure 6.3: Write transaction structure.

6.3 Flit fields Encodings

This subsection describes the different encodings from the channel fields. It includes the field widths and a brief description of each value.

Table 6.3: REQ channel opcodes

Opcode[1:0]	Request command
0x0	Read
0x1	Write
0x2	WritePtl

Table 6.4: Kill value encodings

Kill	Description
0x0	Normal transaction.
0x1	Do not modify memory.

Table 6.5: Excl value encodings

Excl	Description
0x0	Normal transaction.
0x1	Exclusive transaction.

Table 6.6: Attr value encodings

Attr	Description
0x0	Cacheable. Looking up a cache is required.
0x1	Device. Indicates that the memory type associated with the transaction is Device.

6.4 Request and acknowledge handshake

The synchronization mechanism is the same for all the channels. It consists of a valid/ack type of handshake. This handshake transmits information whenever both, valid and acknowledge signals are asserted. Figure 6.4 shows the signal behavior of a data transfer. The number of cycles needed to transmit with this handshake is one.

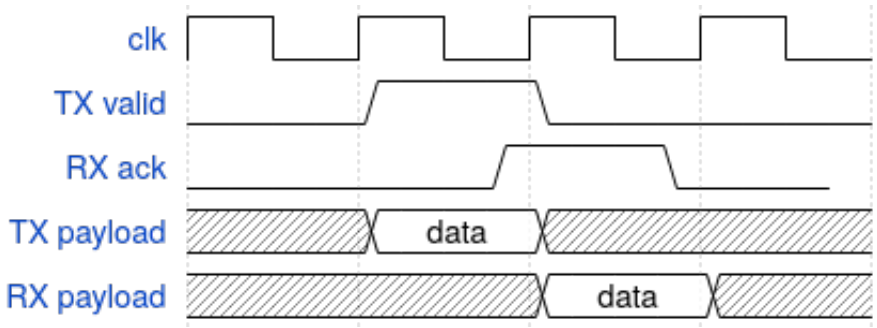


Figure 6.4: Handshake of each channel between VPU and RN.

7 RN-I proposal

This chapter describes the design and verification of an *IO coherent Request Node* based on the AMBA 5 CHI protocol which acts in demand through the interface described in Chapter 6.

It starts by describing the operations it gives support to and how. It goes through the significant AMBA 5 CHI protocol decisions and the overall operation structure chosen.

The design process proposes a top-level data flow followed by a deep dive into all the modules' architecture. Finally, a selection of test cases is analyzed to stress and check the correct behavior of the design once implemented.

Figure 7.1, shows a simple representation of the RN-I port map. Note that the LSU corresponds to the Load/Store Unit from Figure 6.1.

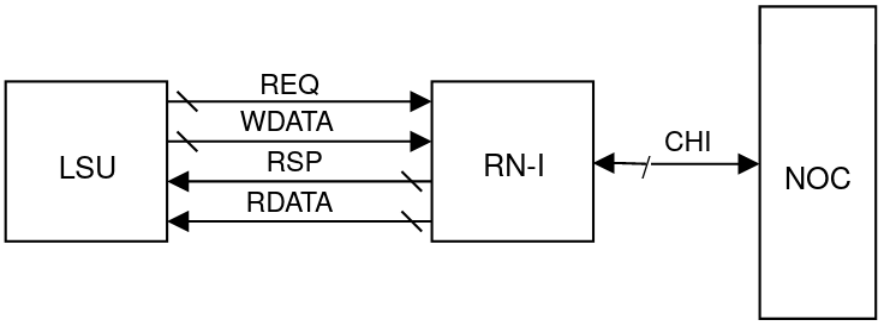


Figure 7.1: Connections of the Request Node.

7.1 Environment constraints

AMBA 5 CHI is a protocol specification rather large and complex. Moreover, key elements inside the NoC , defined in AMBA 5 CHI, are optimized in terms of area and

latency. This optimization brings IP designers to choose carefully which parts of the protocol specification they should include instead of covering the whole specification.

Most of the other modules present in the NoC are already defined when designing this module. In this situation, we should pay particular attention and reuse the already implemented part of the specification whenever possible.

For the NoC of eProcessor, the element that adds these constraints is the host of the second level of cache (L2). This element is a *Home Node Full* with the L2 cache, referred from here on as L2HN. This node has a reduced selection of functionalities compared to the full specification of the HN. Table 7.1 shows the subset of transactions that the L2HN supports and can be used in a newly designed RN-I.

Table 7.1: L2HN Supported CHI opcodes and RN-I comatibles.

Channel	L2HN Opcodes	RN-I Compatibles
Request (RX_REQ)	ReadShared, ReadUnique, ReadOnce, ReadNoSnp, CleanUnique, CleanShared, CleanInvalid, WriteBackFull, WriteCleanFull, WriteUniqueFull, WriteUniquePtl, WriteNoSnpFull, WriteNoSnpPtl, AtomicLoad, AtomicSwap, MakeInvalid	ReadOnce, ReadNoSnp, CleanShared, CleanInvalid, WriteUniqueFull, WriteUniquePtl, WriteNoSnpFull, WriteNoSnpPtl, AtomicLoad, AtomicSwap, MakeInvalid
Data (RX_DAT)	CopyBackWrData, NonCopyBackWrData, SnpRespData, WriteDataCancel	NonCopyBackWrData, WriteDataCancel

7.2 Module operations

This section analyzes the transactions that the module RN-I will need to implement. This set of transactions is deduced from the demands that the RN-I will receive through the interface described in Chapter 6, thus being: Read, Write and Write Partial.

7.2.1 Read

This operation is used to retrieve 512 bits of data. The data retrieved can be from a device memory type or a cacheable location. Exclusive access is also possible for device memory

types. The progress of the read transaction, regardless of their parameters, is as follows:

1. The VPU sends a Read request to the RN-I through the REQ channel.
2. The RN-I sends a Read request to the ICN through the CHI REQ channel with one of the following opcodes:
   - ReadNoSnp
   - ReadOnce
3. The ICN returns to the RN-I the data and any associated transaction response with the CompData opcode on the RDAT channel.
4. The RN-I responds to the VPU with the requested data and the processed error status.

Figure 7.2 shows a read transaction of the RN-I.

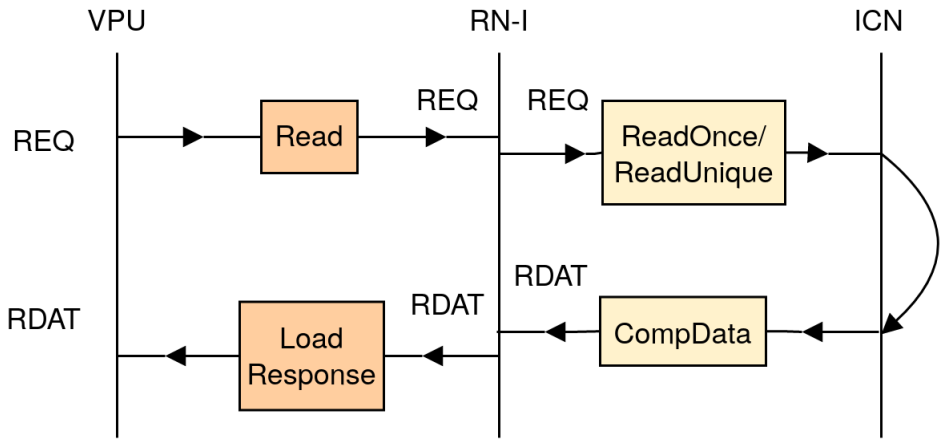


Figure 7.2: Read transaction structure.

7.2.2 Write and WritePtl

These operations can write up to 512 bits, and in the case of WritePtl, they can have byte granularity. The data can be written to a device memory type or a cacheable location. Exclusive access is possible for device memory type. The progress of the write transactions is as follows:

1. The VPU sends a Write request to the RN-I through the REQ channel with one of the following opcodes:
   - Load
   - LoadPtl

2. The RN-I sends a request on the REQ channel with one of the following opcodes:
    - WriteUniqueFull
    - WriteUniquePtl
    - WriteNoSnpFull
    - WriteNoSnpPtl
3. The ICN has the following options:
    - Return two separate responses:
        - Return a DBIDResp response that provides a data buffer identifier indicating that it can accept the write data for the transaction.
        - Return a Comp response to indicate that the transaction is observable by other Requesters.
    - Return a CompDBIDResp response indicating both, (i) it can accept the write data for the transaction, (ii) and that it is observable by other Requesters.
4. The RN-I waits until it has received one of the following responses:
    - The CompDBIDResp associated with the transaction.
    - DBIDResp associated with the transaction.
5. The RN-I tells the VPU, through the RSP channel, that it can send the write data for the transaction.
6. The VPU sends the write data and any associated byte enables to the RN-I or optionally it can cancel the operation with the Kill flag.
7. The RN-I has two options depending on the kill flag:
    - If the Kill flag is asserted, forward the data with the byte deasserted.
    - Otherwise, forward the write data and the associated byte enables.

   Both options are sent with a WriteData opcode through the WDAT channel.

Figure 7.3 shows the write transaction structure of the RN-I.

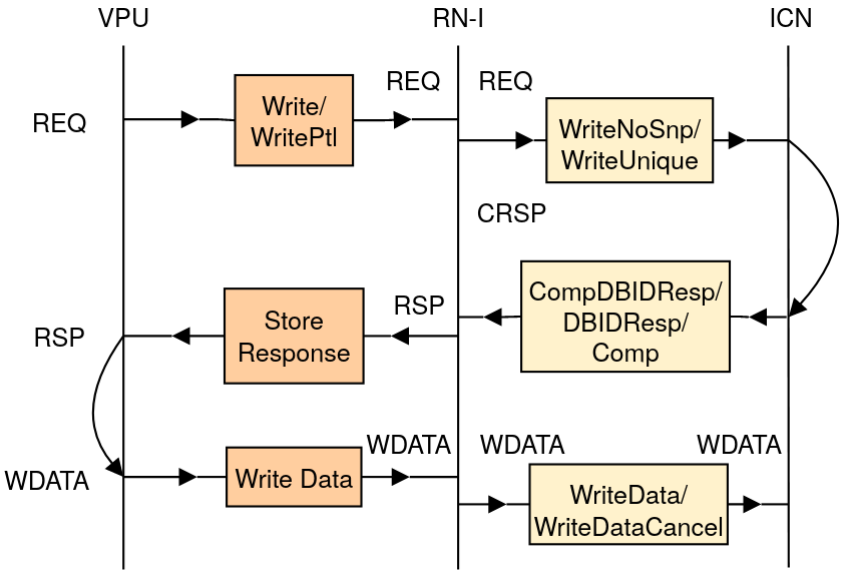


Figure 7.3: Write transaction structure.

7.3 Circuit Design

This section explains the circuit's design process to implement the operations described previously.

Figure 7.4 shows the proposed data path at the top level of the RN-I. This figure uses the acronyms REQ, RDAT, RSP, and WDAT to designate the main modules. Each module is in charge of two channels, one coming from the VPU and another from the ICN. The left module is a lookup table with two read ports, storing information needed to process the incoming messages from the ICN.

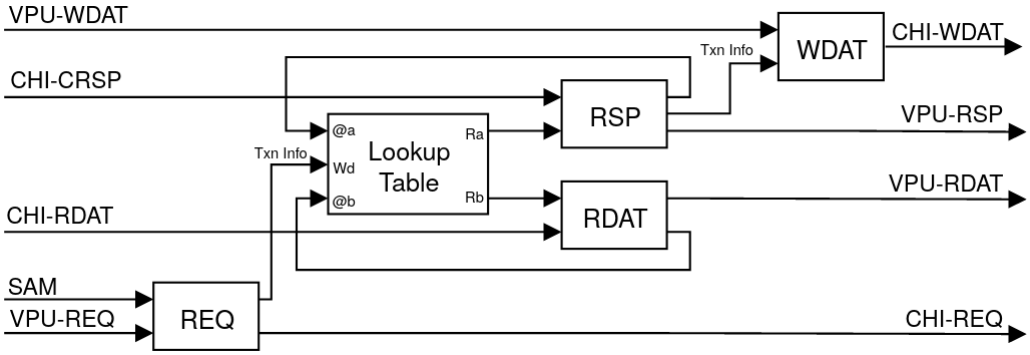


Figure 7.4: Top module layout of the RN-I.

The following sections go through each module involved, describing the internal architecture.

7.3.1 REQ module

The task of this module is to receive a request through the REQ channel of the VPU and forward it to the REQ channel of the ICN in an AMBA 5 CHI compliant way. Figure 7.5 shows the proposed functional block diagram or architecture of the module.

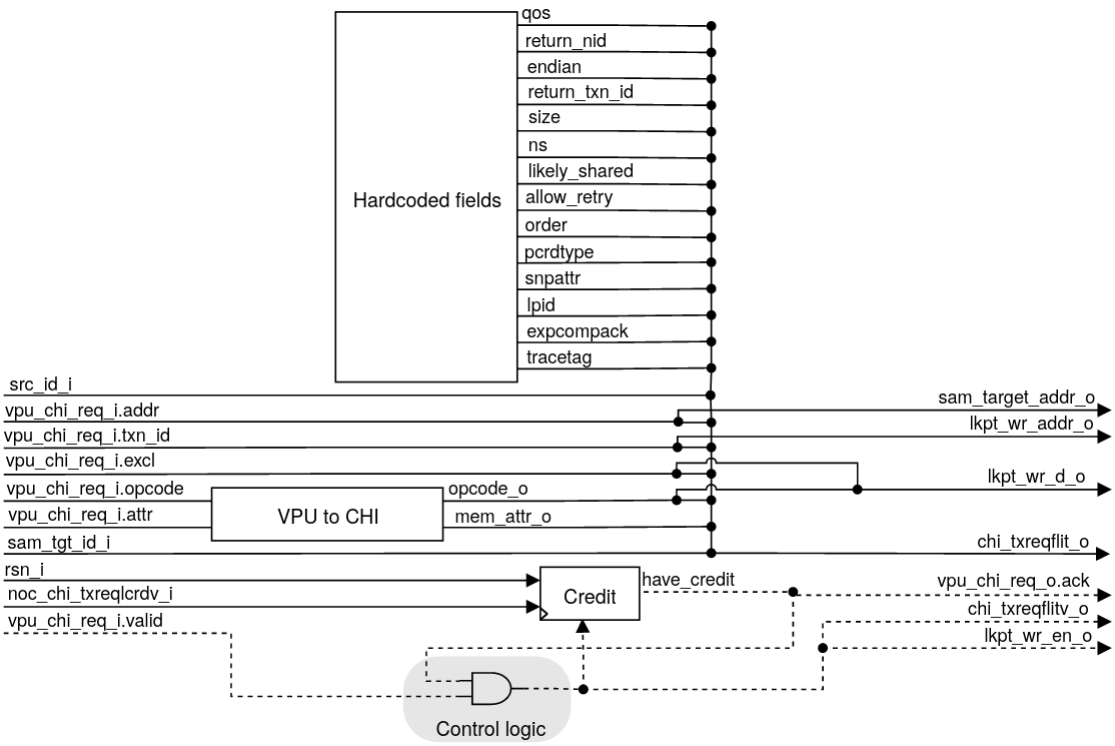


Figure 7.5: REQ module circuit.

Control Logic Forwards the incoming request whenever the VPU requests it, and the sending channel has credit. There is no need to delay the output of the control logic, since the process required to obtain a valid CHI REQ flit is minimal.

Hardcoded fields This block sets all the CHI REQ flit fields that are fixed never change their value. These values can be parameters from the ICN, the VPU, or unused functionalities.

VPU to CHI This block processes the request made by the VPU and determines the Opcode and Attr to be used in the CHI transaction. Table 7.2 defines the mapping process.

Table 7.2: Mapping process made by the *VPU to CHI* block

I/F VPU-REQ		CHI-REQ	
opcode[1:0]	**attr[0:0]**	**Opcode[5:0]**	**MemAttr[3:0]**
Load	0	ReadNoSnp	0010
	1	ReadOnce	0100
Write	0	WriteNoSnpFull	0010
	1	WriteUniqueFull	0100
WritePtl	0	WriteNoSnpPtl	0010
	1	WriteUniquePtl	0100

Credit This block is a submodule already presented in multiple components of the NoC. It implements the CHI credit system, holding the number of credits available into a register and incrementing or decrementing it when necessary. The output of this submodule tells whether the sender can use the channel. Figure 7.6 shows the credit circuit.

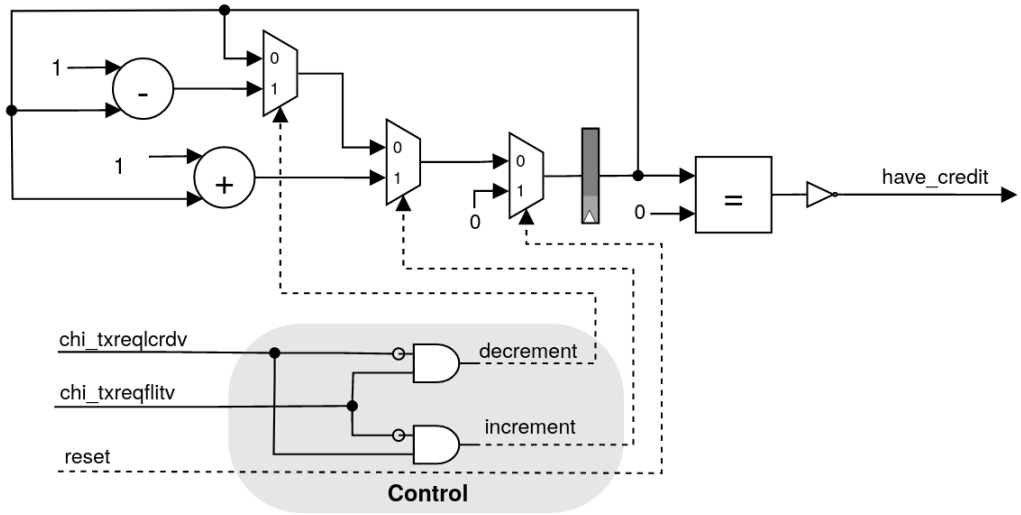


Figure 7.6: Credit module circuit.

7.3.2 RDAT module

This module is in charge of the RDAT channel from CHI and the RDAT channel from the VPU. Figure 7.7 shows the proposed functional block diagram of the module.

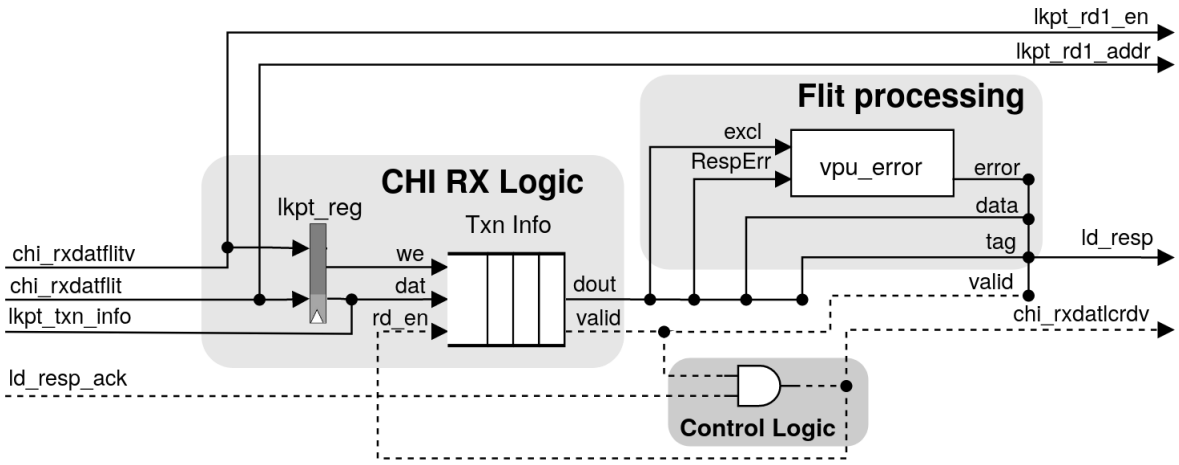


Figure 7.7: RDAT module circuit.

CHI RX Logic This block stores the incoming flit from the ICN along with some necessary information in a FIFO. It has a two-cycle pipeline stage. In the first pipeline stage, it performs a read from the Lookup Table shown in Figure 7.4, and in the second pipeline stage, it writes to the FIFO. The FIFO in use is of type first-word fall-through (FWFT). This type of FIFO always outputs the first element of the queue with the valid signal asserted, if it is not empty. The size of the FIFO is the maximum number of credits that the receiver can provide. This parameter is defined across the ICN. Each entry of the FIFO stores the Excl bit read from the Lookup Table with the following flit fields:

- RespErr
- Data
- TxnID

Flit processing This block is in charge of processing the incoming CHI flit and producing the corresponding load response for the VPU. Its only processing task consists of interpreting the CHI field *RespErr* according to whether the transaction was exclusive or not.

Control Logic The control logic of the module handles the progress of the received flits from the ICN to the VPU. Whenever the VPU receives a load response, it removes the entry from the FIFO and sends a credit through the CHI channel.

7.3.3 RSP module

The task of this module is to receive flits from the CHI-RSP channel, process them, and propagate the result to the WDAT module and the VPU-RSP channel. The architecture of this module is really similar to the RDAT module, because both perform a similar task. The main difference is an additional link with the WDAT module used to propagate information for write transactions. Figure 7.8 shows the proposed functional block diagram of the module.

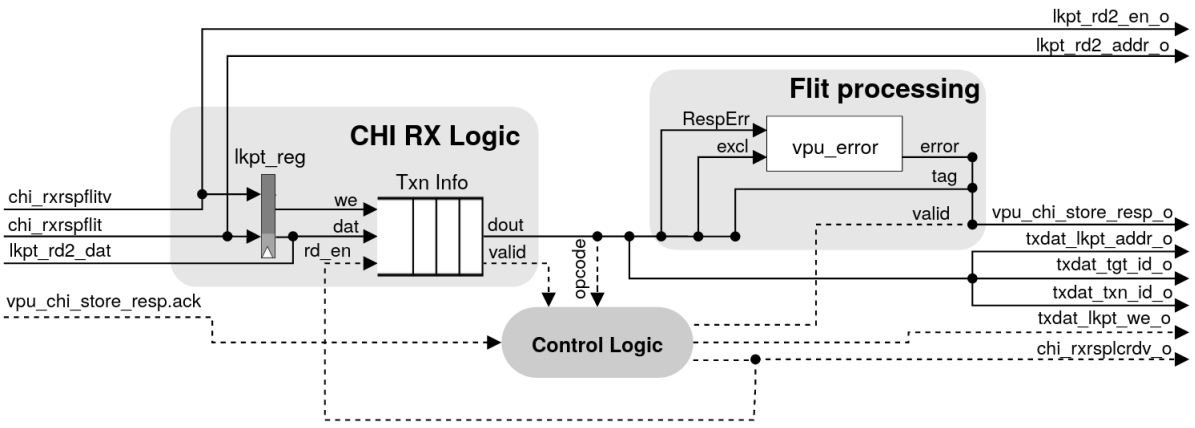


Figure 7.8: RSP module circuit.

Control Logic As described in Section 7.2.2, this module can receive three types of responses. Despite this, it is only needed to process: *DBIDResp* and *CompDBIDResp*. The rest can be dropped without any further action. Once one of these two responses is received, the module performs the following two actions:

- Send a store response to the VPU indicating that it can send the data to write.
- Inform the WDAT module about the TxnID and TgtID for the *WriteData* packet.

Figure 7.9 shows the Control Logic, which is centralized. It differentiates in the part corresponding to the detection of flits to process and the part corresponding to the actuation.

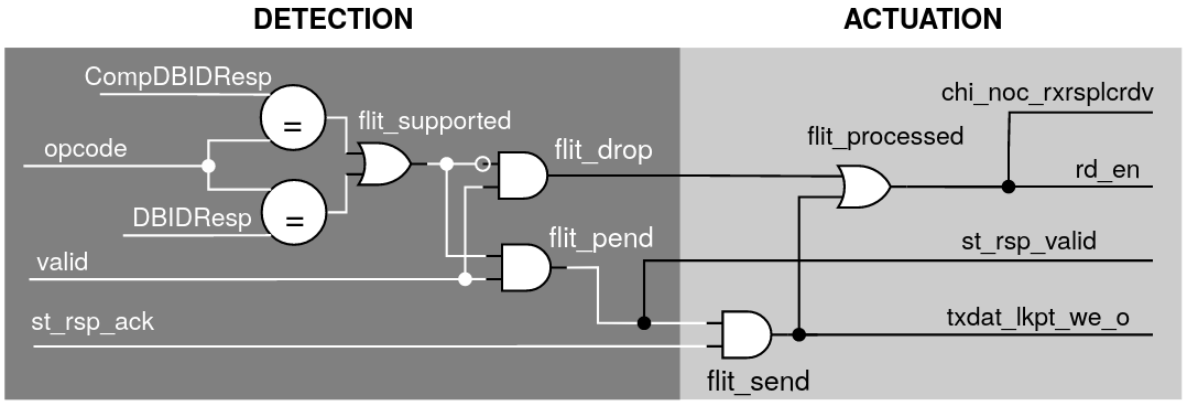


Figure 7.9: Control Logic from the RSP module.

7.3.4 WDAT module

This module receives the store data from the VPU, the transaction information from the module RSP builds the corresponding flit and sends it through the WDAT CHI channel. Figure 7.10 shows the proposed functional block diagram of the module.

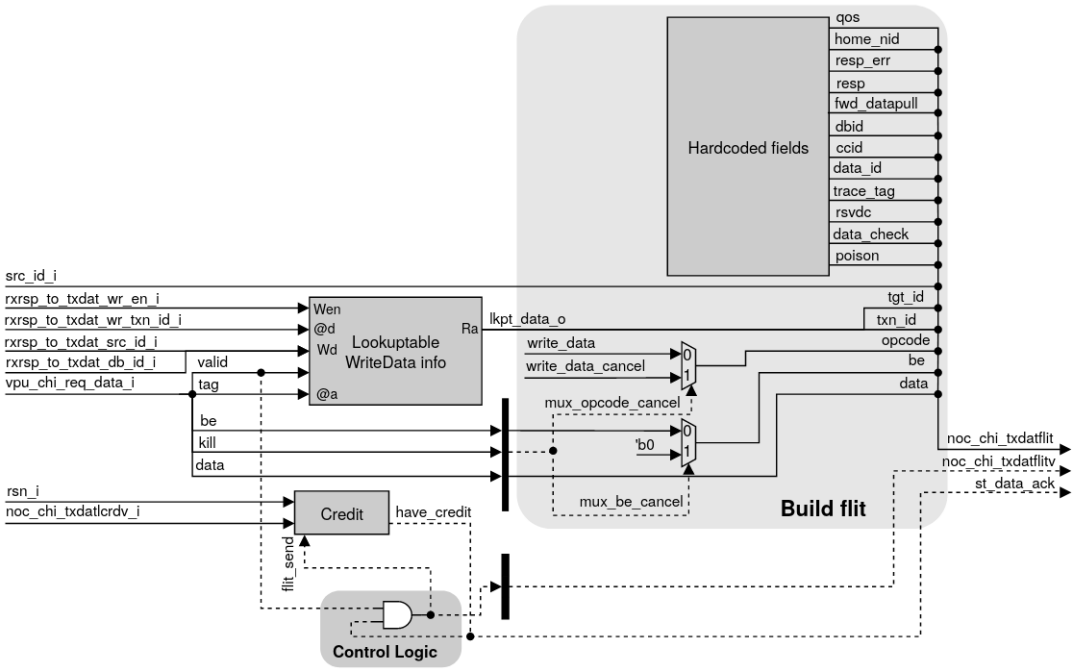


Figure 7.10: WDAT module circuit.

Lookup Table WriteData info Stores the transaction information obtained from other modules. The stored transaction information consists of the DBID and SrcID from the RSP module and they are used as TxnID and TgtID of the WDAT flit, respectively. The index used to access the information of the table is the Tag field from the interface with the VPU described in Chapter 6.

Credit This module is already explained in Section 7.3.1.

Build flit This block is in charge of compiling all the information needed to build a CHI-WDAT compliant flit. The required data at this module comes from the VPU-WDAT channel and the Lookup Table. The build process can prepare a flit to perform the write operation or to cancel it,23 depending on the Kill flag. In case of cancellation, the byte enable is deasserted, and the Opcode is set to WriteDataCancel.

Control Logic This control logic is the same as the one explained in REQ module. The only difference is that the propagation of the *txdatflitv* signal is not forwarded to the ICN directly. In this case, an additional read operation to the Lookup Table is performed, which adds a cycle delay to the signal.

7.3.5 Transaction lookup table

This table stores transaction information. The information stored is written as soon as a new request arrives at the RN-I. The primary use of this information is to process incoming flits from the ICN. The modules that have this requirement are RSP and RDAT. This setup is accomplished with one write port controlled by the REQ module and two read ports used by the RSP and RDAT modules.

The information stored in the lookup table is the excl flag associated with each transaction. It is later used to parse the CHI flit RespErr.

In the future, the transaction information stored in the table will likely grow as more functionalities get added to the interface described in Chapter 6.

7.4 Verification

This section proposes a set of tests derived from the design and implementation of the RN-I. These tests are divided into two groups. The first group focuses on testing the

RN-I's design, and the second on integrating it with the NoC. The tests discussed in this section were simulated with *QuestaSim* 2019.4, using a behavioral simulation model of the Register Transfer Level (RTL).

The simulation environment is composed of three elements. These elements are the already mentioned RN-I as the Design Under Test (DUT), the NoC, and a non-synthesizable module that emulates the VPU. The NoC corresponds to the one used in the EPI simulation model. The AMBA 5 CHI components are HN-F, SN-F, and the remaining ICN units, such as routers. Moreover, the ICN topology of the simulation model is a configurable-size mesh. The NoC includes a monitoring system for all the AMBA 5 CHI channels. The monitoring system provides a in-depth view of the messages sent from each node. Figure 7.11 illustrates the ICN topology configured as a 4x4 mesh.

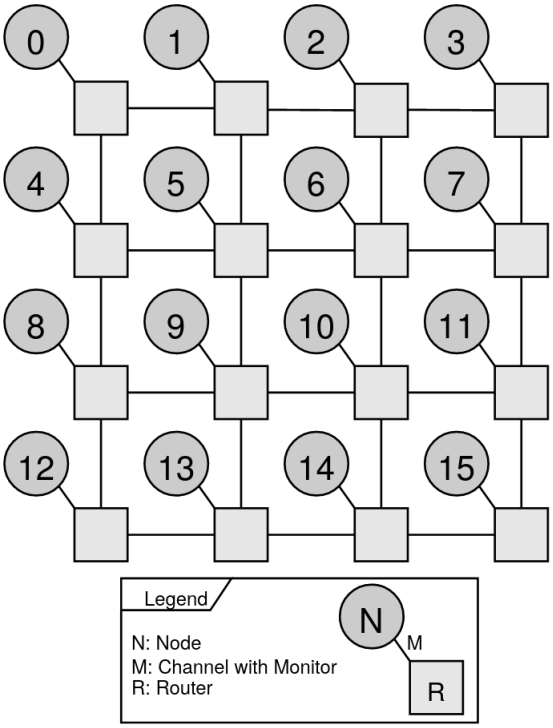


Figure 7.11: Mesh network configured as 4x4.

The monitor system registers any flit sent, including source, destination, and content. Listing 7.1 shows the RN-I messages logged from the RN-I channels in a load operation.

Listing 7.1: Loging of the ICN channels from the RN-I in a load operation

```
50030000:rni(1) txn(1) sends ReadOnce req flit to ...
50030000:rni(1) txn(1) sends req flit: qos ( 0 )  ...
50030000:rni(1) txn(1) sends req flit: allowretry ...
51630000:rni(1) txn(1) gets COMP_DATA_I dat flit  ...
51630000:rni(1) txn(1) gets dat flit: qos ( 0 ),  ...
51630000:rni(1) txn(1) gets dat flit: dbid ( 1 ), ...
```

7.4.1 RN-I tests

The testing strategy consists of the following steps:

1. Modules are divided into functionalities and associated with logic blocks.
2. The test produces all legal combinations of the block's input signals.
3. It checks a condition that, when passed, indicates the correct behavior of the block.

The test scenario is specially configured to be as light as possible to reduce simulation time. This environment is composed of the smallest mesh, populated with two RN-Is, one HN-F, and one SN-F containing the memory. The tests are carried out with an additional non-synthesizable module connected to one of the RN-Is through the interface described in Chapter 6. This module will stimulate the RN-I to produce the appropriate test patterns. Figure 7.12 shows the defined layout of the mesh.

The base test consists of a sequence of stores to a region of memory followed by a series of loads retrieving it. The test passes if the data retrieved matches the data stored. The following test cases proposed focus on corner cases since the base test already verifies the correctness of most parts of the design.

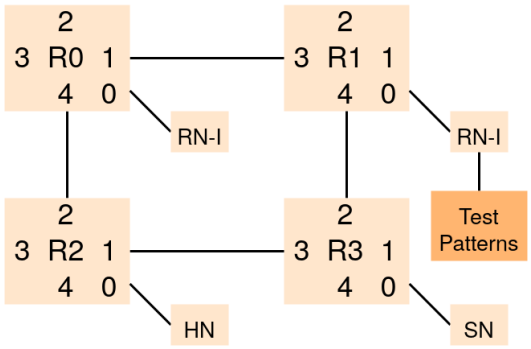


Figure 7.12: 2x2 mesh used to test the RN-I.

The blocks identified and their test cases are described below.

REQ Module In this module, it is possible to identify three blocks to test:

- **Control Logic:** The test generates a specific corner case, where the CHI channel runs out of credit. This case can be reached by maximizing the throughput of the VPU-REQ channel and slowing down the rest.
- **VPU to CHI:** The test generates transactions with all the possible Opcodes and Attr. Table 7.2 shows the complete list of combinations.
- **Credit:** This submodule was not developed in this thesis and is already verified.

RDAT Module In this module, it is possible to identify three blocks to test:

- **CHI RX Logic:** The base test verifies this logic.
- **Flit Processing:** The input information of this block comes from the ICN, thus not allowing the test environment to efficiently stimulate the signals. The easiest way to verify it is to simulate a simple setting where the block is a stand-alone module.
- **Control Logic:** The test generates an corner case that has a low receiving rate.

RSP Module In this module, it is possible to identify three blocks to test:

- **CHI RX Logic:** The base test verifies this logic.
- **Flit Processing:** The functionality from this block is already tested in RDAT module.
- **Control Logic:** Some input signals of the block come from the ICN, which limits the capabilities of the test environment. Despite this, the test lowers the receiving rate from the VPU-RSP channel and traces ICN messages to ensure that the module has dealt properly with *CompDBIDResp*, *DBIDResp*, and *Comp*.

WDAT Module In this module it is possible to identify three blocks to test:

- **CHI RX Logic:** This logic is tested in the base test.
- **Build flit:** The test generates a sequence of stores where some of them use the cancellation mechanism.

- **Control Logic:** The test generates different sending rates to make the CHI channel run out of credit. To maximize the throughput, the VPU-WDAT channel stalls until a big enough queue of messages is ready to be sent. Once the condition is met, it starts sending them, one per cycle. This pattern encounters the capacity limitation of the HN, which sets the maximum transactions that can be stopped at this point.

The test results showed that some bug fixes were needed across the design. Despite this, no major flaws were encountered with the proposed design. Some of the fixes applied consisted of:

- Ensuring correct management of the clock signal.
- Removal of some unnecessary CHI flit fields.
- Fixing some logic from the RSP Control Logic.

7.4.2 NoC with RN-I tests

This group of tests is intended to stress the NoC with the newly designed node. These tests lay out different environments and stimulate the RN-Is to use the NoC at its limit.

For this group of tests, a special test pattern module is developed. Its goal is to produce requests with address ranges and data sets that do not overlap. This goal is accomplished by providing an additional parameter to the test pattern that uniquely identifies it. The parameter is then used to calculate the offset of the range. The same principle is used with the data set.

Multiple RN-Is In this environment both RN-Is have a Test Pattern module connected to them. The mesh is the same as the one used previously.

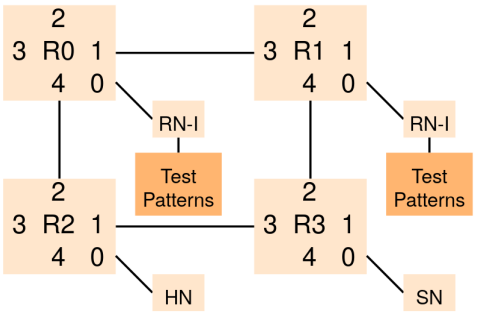


Figure 7.13: 2x2 mesh with two acting RN-Is.

Multiple HN This environment uses a 4x4 mesh. It features sixteen RN-Is, sixteen HNs, and four SNs. Only two of the RN-Is have Test Pattern modules connected to them. The mesh image has been omitted for conciseness.

The test results showed a number of problematic cases. These problems concerned different node types and were reported to the NoC maintainers.

Slave Node Two issues were encountered. The initialization process was managed in such a way were the transactions received during that time where dropped without any response or warning. The generation of DBID did not take into account multiple RN-Is, thus mixing incoming transactions with the same TxnID, but from different sources.

Home Node Some opcodes used by the RN-I were not implemented.

8 Conclusions

This thesis aimed to improve the memory path of the VPU, an accelerator in eProcessor. This objective has been accomplished with a new memory path that reduces latency by allowing the VPU to directly have access to the second level of cache (L2), rather than its predecessor. The final path includes a request node for the NoC connected to the VPU through a custom interface.

The obtained result opens up a range of possibilities for the VPU, as its memory path does not go through the processor core anymore. The new interface also hides details of the AMBA 5 CHI protocol from the NoC to allow greater flexibility in implementing the VPU.

The AMBA 5 CHI Request Node, described in Chapter 7, is specially designed to propagate operations coming from the VPU. It implements reads, writes, and partial writes in an AMBA 5 CHI-compliant way. It also implements a cancellation mechanism that allows the VPU to perform speculative memory operations. The design uses a modular approach that facilitates new features in the future. Another positive aspect of the Request Node is that, since it only serves the VPU, it is fully adaptable to its demands.

Based on the analysis of AMBA 5 CHI performed during the design and implementation of both the interface and the Request Node, it can be concluded that the new memory path will easily meet its expectations in a mid to long-term future. The AMBA 5 CHI protocol offers many options and functionalities yet to be used and experimented with, such as the *Early Write Acknowledgement* (EWA).

8.1 Future Work

Even though the proposed memory path achieved the objectives of this thesis, some more steps are still needed to be done in terms of architecture, integration, and development environment.

1. The current design of the RN-I has a peculiarity regarding the channels where it acts as a transmitter (TX). This peculiarity consists of each channel having a single source event that causes a flit to be sent. For example, the source event that causes a flit to be sent through the CHI-REQ channel is an incoming request from the VPU-REQ channel. As functionalities get added, more source events will appear. Continuing with the CHI-REQ channel example, if the RN-I were to implement the retry mechanism, an incoming flit from the CHI-RSP channel could also be a source event that causes a CHI-REQ flit to be sent. The described situation brings the need to arbitrate resources like the CHI-REQ channel. With all this said, a future architectural task will be the addition of arbiters to the resources that encounter structural risks and adapt the modules' control logic, ensuring good forward progress for the transactions.

2. When finishing this thesis, the integration tests verified the RN-I model together with the AMBA 5 CHI NoC. However, the VPU RTL model is not yet completed, thus preventing including it in the process. This situation forces us to leave the integration of the RN-I, the VPU, and the NoC as a future task.

3. Finally, the simulation environment used to produce and evaluate test cases does not scale comfortably. It is highly encouraged for future work to develop a more sophisticated way of testing that further exploits the verification features available by the technologies used to implement the models. This set of features could include constrained randomization, assertions, coverage support, or even consider using Universal Verification Methodology (UVM).

References

[1] Anysylicon, "Understanding AMBA Bus Architechture and Protocols," [Online] .Available:https://anysilicon.com/understanding-amba-bus-architecture-protocols/, Accessed on 2021-09-28.

[2] *AMBA 5 CHI Architecture Specification*, Arm Limited or its affiliates, 5 2018, rev. C.

[3] digital-strategy, "European Processor Initiative: consortium to develop Europe's microprocessors for future supercomputers," [Online].Available:https://digital-strategy.ec.europa.eu/en/news/european-processor-initiative-consortium-develop-europes-microprocessors-future-supercomputers, Accessed on 2021-09-28.

[4] ——, "The European declaration on High-Performance Computing," [Online].Available:https://digital-strategy.ec.europa.eu/en/news/european-declaration-high-performance-computing, Accessed on 2021-09-28.

[5] ——, "The European High Performance Computing Joint Undertaking," [Online].Available:https://digital-strategy.ec.europa.eu/en/policies/high-performance-computing-joint-undertaking, Accessed on 2021-09-28.

[6] bsc.es, "EPI: European Processor Initative (EPI)," [Online].Available:https://www.bsc.es/research-and-development/projects/epi-european-processor-initative-epi, Accessed on 2021-09-28.

[7] Thierry Breton, "How a European Chips Act will put Europe back in the tech race," [Online].Available:https://ec.europa.eu/commission/commissioners/2019-2024/breton/blog/how-european-chips-act-will-put-europe-back-tech-race_en, Accessed on 2021-09-28.

[8] eprocessor.eu, "About," [Online].Available:https://eprocessor.eu/about/, Accessed on 2021-09-28.

[9] wikipedia.org, "RISC-V," [Online].Available:https://en.wikipedia.org/wiki/RISC-V, Accessed on 2021-09-28.

[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach Fifth Edition*. Elsevier, 2012.

[11] arm.com, "AMBA 5 Overview," [Online].Available:https://developer.arm.com/architectures/system-architectures/amba/amba-5, Accessed on 2021-09-28.

[12] R. Espasa and P. Marcuello and A. Moreno and S. Pomata, "AVISPADO - VPU Interface," [Online].Available:https://semidynamics.com/technology, Accessed on 2021-09-28.

[13] jobted.es, "Sueldo del Ingeniero Informático en España," [Online].Available:https://www.jobted.es/salario/ingeniero-inform%C3%A1tico, Accessed on 2021-09-28.

[14] intel.com, "Buy Modelsim," [Online].Available:https://buyfpga.intel.com/PartDetail?partId=2021454, Accessed on 2021-09-28.

[15] elpais.com, "Herramientas financieras," [Online].Available:https://cincodias.elpais.com/herramientas/calculadora-sueldo-neto/#tabla_resultados, Accessed on 2021-09-28.

[16] five-embeddedv, "14 RVWMO Memory Consistency Model, Version 2.0," [Online].Available:https://five-embeddev.com/riscv-isa-manual/latest/rvwmo.html, Accessed on 2021-09-28.